CASSETTE D'ACCOMPAGNEMENT DISPONIBLE

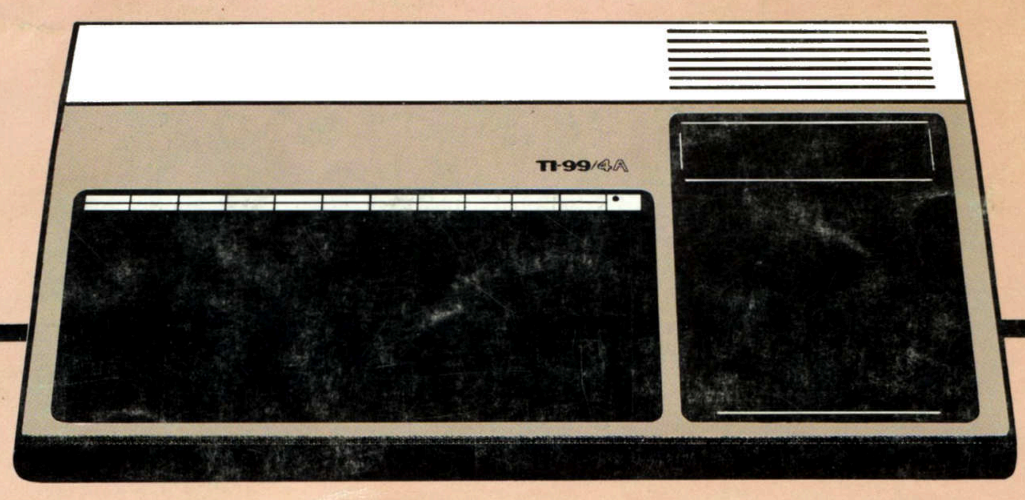
99 MAGAZINE

GRAPHISME EN HAUTE-RESOLUTION

BASIC, PASCAL, LOGO, ASSEMBLEUR...

DES PROGRAMMES POUR VOTRE TI

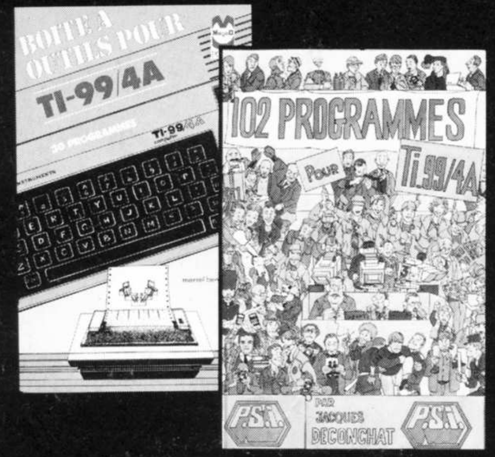
UTILISATION DE LA MINI-MEMOIRE



NUMÉRO 4 - Trimestriel - Mars 1984 - 40 F

ISSN 0759-9005

SI(X) TEXAS m'étaient contés...



... 6 livres P.S.I. pour votre TI-99/4A

Nouveautés

BOITE A OUTILS POUR TI-99/4A *par Marcel Henrot*

Ces livres de poche n'ont pas la prétention de livrer le secret des grandes applications informatiques, mais bien de mettre à la disposition des utilisateurs des petits programmes tout prêts qui leur permettront de résoudre de nombreux problèmes de la vie quotidienne. La Boîte à Outils pour TI-99/4A vous permettra d'élaborer des dessins cubiques, d'agrandir un caractère, de transformer l'ordinateur en prof de maths, de tracer une ellipse...
Format : 11,5×18 - 128 pages - **35,00 FF / 250 FB / 12,20 FS**
Port : 10 FF / 50 FB / 1,50 FS.

102 PROGRAMMES POUR TI-99/4A *par Jacques Deconchat*

Apprendre en se distrayant, tel est l'objectif de ce livre. Ces 102 programmes, classés par niveau vous permettront de découvrir l'ordinateur individuel TI-99/4A et d'acquérir progressivement des connaissances nouvelles et une plus grande maîtrise du Basic. La description des jeux proposés (jeu de l'oie aléatoire, saute-mouton, dé truqué, mastermind, échiquier, pendu...) est suivie d'un programme commenté et d'un exemple d'utilisation.
Format : 25×17 - 240 pages - **110,00 FF / 850 FB / 34,60 FS.**

LA DECOUVERTE DU TI-99/4A

par Frédéric Lévy et Dominique Schraen
Une approche progressive et attrayante pour les débutants qui veulent apprendre la programmation en Basic du TI-99/4A. Ils commenceront par apprendre les instructions classiques du Basic telles qu'elles existent sur le TI-99/4A. Ils découvriront ensuite les possibilités graphiques et musicales de cet ordinateur grâce à de nombreux exemples et exercices.
Format 14,5 × 21 - 144 pages - **90 FF / 695 FB / 28,40 FS.**

EXERCICES POUR TI-99/4A

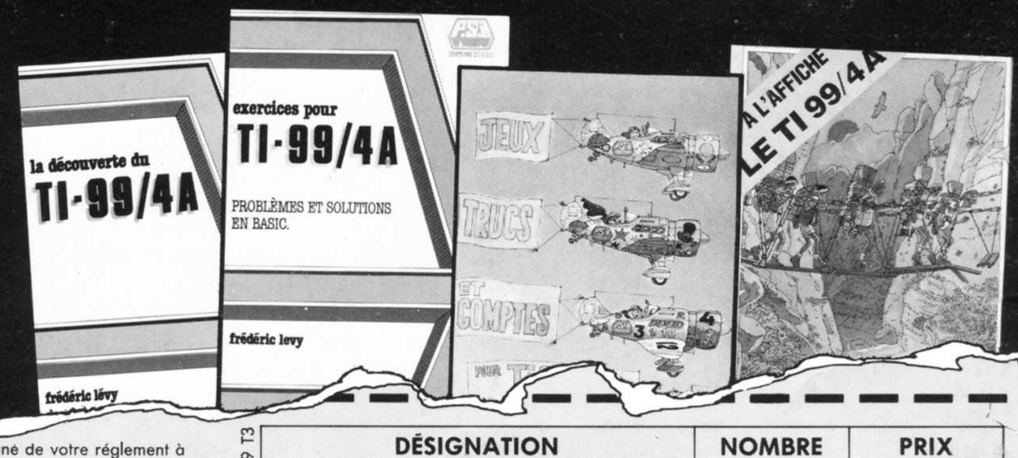
par Frédéric Lévy
Ce livre est destiné à tous ceux qui connaissent les instructions Basic du TI-99/4A et qui ne maîtrisent pas encore la programmation. Ce recueil d'exercices est une invitation à l'analyse et à la programmation de problèmes simples et fréquemment rencontrés (calculs, tris, traitement de texte...). L'énoncé de chaque exercice est suivi de son analyse, une ou deux solutions commentées sont proposées.
Format 14,5 × 21 - 144 pages - **90 FF / 695 FB / 28,40 FS.**

JEUX, TRUCS ET COMPTES POUR TI-99/4A

par Michel Benelfoul.
Que programmer ? Comment ? Pour quoi faire ?... « Jeux, trucs et comptes pour TI-99/4A » propose aux novices de la programmation 30 programmes en TI-Basic commentés et décrits à l'aide d'un exemple d'exécution et d'un organigramme. Fonctions TI-Basic, jeux passifs et interactifs, interludes, « trucs », paie, facturation simple et routines, l'ensemble de ce recueil très varié leur permettra d'utiliser au mieux toutes les fonctions de leur TI-99/4A.
Format 17 × 25 - 176 pages - **90 FF / 695 FB / 28,40 FS.**

LE TI-99/4A A l'AFFICHE

par Jean-François Sehan
Cet ouvrage, qui s'adresse aux possesseurs de TI-99/4A, regroupe 20 programmes à caractère utile, familial ou ludique. Du papier peint à la cuisine en passant par le Baccara et la fête foraine, chaque programme est accompagné d'un organigramme, d'une liste de variables et d'une explication de chaque ligne Basic, pour l'adaptation éventuelle à d'autres ordinateurs.
Format 17 × 25 - 144 pages - **90 FF / 695,00 FB / 28,40 FS.**





Envoyer ce bon accompagné de votre règlement à
P.S.I DIFFUSION
B.P. 86 - 77402 Lagny-s/Marne Cedex - FRANCE
Téléphone : (6) 006.44.35

ou pour la Belgique et Luxembourg à :
P.S.I. BENELUX
5, Av. de la Ferme Rose - 1180 Bruxelles BELGIQUE
Téléphone : (2) 345.08.50

99 T3

DÉSIGNATION	NOMBRE	PRIX
par avion ajouter 8 FF (75 FB) par livre	**TOTAL**	

en Suisse : P.S.I. SUISSE
Case Postale - Route Neuve 1 - 1701 Fribourg SUISSE
Téléphone : (037) 23.18.28 - C.C.P. : 17.56.84

au Canada : S.C.E. INC.
65, Avenue Hillside - Montréal (Westmount) Québec H 3 Z 1 W 1 - Tél. : (514) 935.13.14

Nom : ____________ Prénom : ____________
Rue : ____________ N° ____
Code Postal : ______ Ville : ____________

☐ *Paiement par chèque joint*
☐ *Paiement en FF par carte bleu VISA* N° ____________ Date d'expiration ______
(pour la France et à P.S.I. Diffusion exclusivement)

Signature (obligatoire pour paiement par carte de crédit)



Sommaire

Editions MEV - 49, rue Lamartine - 78000 Versailles - Tél. : (3) 951.24.43

(Dépôt légal : 1er trimestre 1984)

Editorial

Ce numéro de « 99 Magazine » est plein à craquer ; il a donc été nécessaire de modifier la mise en page de certains listings afin que tous les articles et programmes puissent être publiés.

Vous êtes nombreux à ne pas pouvoir vous procurer le module « Basic Etendu » ; nous avons donc décidé d'augmenter le nombre de programmes en Basic TI. Cependant, « 99 Magazine » est la revue de TOUS les utilisateurs du TI-99 et les « mordus » de Pascal, Assembleur, Logo ou Basic Etendu trouveront dans ce numéro de quoi occuper leurs loisirs.

En outre, vous êtes de plus en plus nombreux à nous faire le plaisir de lire cette revue, ce qui renforce notre détermination à continuer la publication de « 99 Magazine ».

*Dans ce numéro, vous trouverez de nombreux programmes : **Michel Teyssou** vous transforme en pilote d'hélicoptère, **Edouard Krok** vous propose un stage de serveur de restaurant, pendant que **Jean-Claude Hoet** vous entraine dans un combat de chars.*
***Jean-Claude Cornevin** met votre mémoire visuelle à l'épreuve alors que **Marianne Sutz,** avec « Jack-Pot », met votre fortune en situation périlleuse. **Christian Monfort** transforme votre TI en horloge, **Olivier Colin** vous propose une partie de « Solitaire » et **Julien Thomas** vous présente une nouvelle production du club TISOFT-HOME : « Claustrophobia ».*
*Au chapitre des programmes il faut aussi signaler un chargeur automatique en Basic Etendu, écrit par **Eric Thomas.***

***Denise Amrouche** vous indique comment utiliser la Mini-mémoire sans savoir programmer en Assembleur pendant que **Roger Didi** poursuit sa série d'articles sur l'apprentissage du Basic.*
***Gérard Santraille** et **Jean-Luc Bazanegue** implantent de nouvelles fonctions graphiques sur votre TI à l'aide de routines en Assembleur. **Gérard** vous propose aussi un article sur le Système P-UCSD alors que **Jean-Luc** vous explique le fonctionnement des sous-programmes en Assembleur.*
*En outre, **Julien Thomas** débute, avec les lutins, une série d'articles sur les particularités du Basic Etendu pendant que **Sophie** vous fait découvrir comment Logo manipule les nombres.*

*Pour les bricoleurs : **Guy Fix** vous communique des renseignements qui vous permettront de réaliser vous-même un cordon de magnétophone ou de transformer votre TI en détecteur de lumière. Vous trouverez aussi dans ce numéro plusieurs trucs et astuces.*

Hervé Thiriez

Rédacteur en chef - Directeur de la publication : Hervé Thiriez. **Comité de rédaction :** Jean-Luc Bazanegue, Thomas Coppens, Roger Didi, Gérard Santraille. **Dessins :** Laurent Bidot. **Ont collaboré à ce numéro :** Denise Amrouche, Jean-Luc Bazanegue, Olivier Colin, Jean-Claude Cornevin, Roger Didi, Alexandre Duback, Sophie Ehster, Guy Fix, Jean-Claude Hoet, Edouard Krok, Christian Monfort, Alain Paloma, Gilles Pavy, Gérard Santraille, Sophie, Marianne Sutz, Michel Teyssou, Robert Thibodeau, Eric Thomas, Julien Thomas, Eric Woerner.

Editions MEV - 49, rue Lamartine - 78000 Versailles. Tél. : (3) 951.24.43.

Régie publicitaire : Force 7, Anne Jourdan, 39, rue de la Grande-aux-Belles, 75483 Paris Cedex 10. Tél. : (1) 238.66.10.

Diffusion auprès des boutiques et libraires : Editions du PSI, 44-51, rue Jacquard, BP 86, 77400 Lagny-sur-Marne.

Composition : Télécompo, 13-15, avenue du Petit Parc, 94300 Vincennes.

Impression : Imprimerie Rosay, 47, avenue de Paris, 94300 Vincennes. Tél. : (1) 328.18.63.

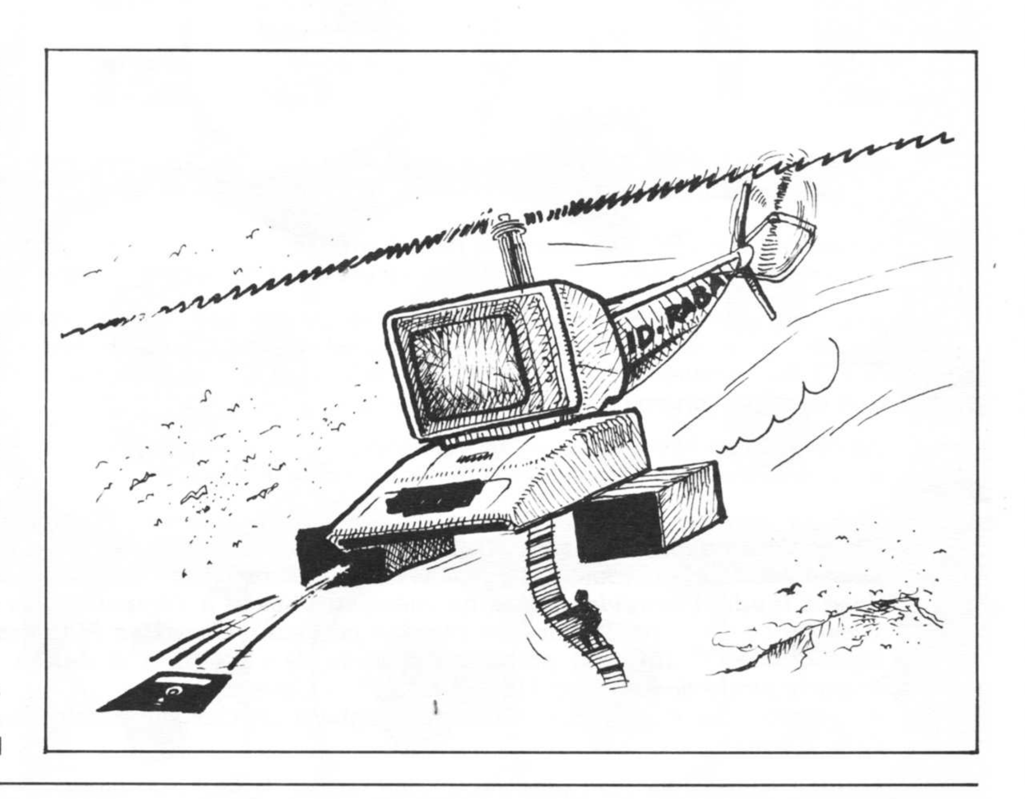
S.O.S Hélicoptère

Michel Teyssou

Trois alpinistes sont bloqués par le mauvais temps au sommet d'une montagne. Votre mission consiste à aller les récupérer avec un hélicoptère. Vous devez donc partir de la vallée et vous diriger vers le sommet de la montagne, sur lequel vous devrez vous poser. Pour diriger votre appareil, utilisez les touches fléchées.

L'hélicoptère n'est pas d'un modèle très récent et les commandes ne réagissent pas toujours très bien; une parfaite maîtrise de l'appareil demande un peu d'entraînement. De plus, la visibilité est très mauvaise : des nuages défilent constamment sur la montagne.

Lorsque vous aurez récupéré un alpiniste (l'hélicoptère n'accepte qu'un passager), il faudra le ramener dans la vallée, si possible en bon état.
Un petit conseil : si vous avez les nerfs fragiles, n'utilisez pas ce programme écrit en **Basic Etendu**. ■



```
100 ! ****************
110 ! *  SOS HELICO  *
120 ! ****************
130 ! * BASIC ETENDU *
140 ! ****************
150 ! *  COPYRIGHT   *
160 ! *MICHEL TEYSSOU*
170 ! *      ET      *
180 ! * 99  MAGAZINE *
190 ! ****************
200 DATA 392,1,392,3,392,1,523,4,523,4,58
7,4,587,4,784,7,659,2,523,4,392,30 !MUS
IQUE
210 CALL CLEAR
220 CALL SCREEN(5)
230 CALL COLOR(1,2,2)
240 GOSUB 2960
250 CALL COLOR(14,16,2)
260 A$=RPT$("0",12)&"02000000A14FA1"&RPT$
("0",18)&"AA0070C8E4F2BECC78FC" !HELICO
270 AA$=RPT$("0",12)&"AA000E13274F7D331E3
F"&RPT$("0",12)&"A000000082F582" !HELIC
O
280 B$=RPT$("0",12)&"0500000041AF41"&RPT$
("0",18)&"550070C8E4F2BECC78FC" !HELICO
290 BB$=RPT$("0",12)&"55000E13274F7D331E3
F"&RPT$("0",12)&"4000000085F285" !HELIC
O
300 C$="99423C1818181818" ! BONHOMME RAME
NE
310 D$="42427E4242427E42" ! ECHELLE
320 T$="18187E18183C7E7E" ! TOMBE
330 E$="0102040810204080" ! DESSIN
340 F$="8040201008040201" ! DESSIN
350 G$=RPT$("01",8)!DESSIN
360 H$=RPT$("80",8)!DESSIN
370 I$="FFFF424242FFFF" !   BASES
380 X$="000F3F7FFFFF3FBF9FC3FFFF7F3F0F000
0E0F8FEFFFFFFFFFFEF8F0F0E0C080" !NUAGES
390 Y$="8903271F9E1C39F83F1E9C1D270389002
280C87072F0F83E3870F2F0C88022" !EXPLOSI
ON
400 J$="0000010100070B112001020202020206000
000808000E0D0880480404040404060" !ALPINIS
TE
410 K$="00002111080703010001020202020206000
000848810E0C08000804040404040060" !ALPINIS
TE
420 L$="0000010100472B11010F080818000000000
008888810E0C080808040404040404060" !ALPINIS
TE
430 M$="0010111118070301010102020202020206000
000808000E2D48880F0101018" !ALPINISTE
440 CALL DELSPRITE(ALL)
450 FOR I=2 TO 11 :: CALL COLOR(I,16,2)::
 NEXT I
460 CALL COLOR(12,12,2)
470 CALL CHAR(56,BB$)!HELICO
480 CALL CHAR(60,AA$)!HELICO
490 CALL CHAR(136,A$)!HELICO
500 CALL CHAR(140,B$)!HELICO
510 CALL CHAR(128,J$)!ALP.
520 CALL CHAR(132,K$)!ALP.
530 CALL CHAR(120,E$)!DESSIN
540 CALL CHAR(121,F$)!DESSIN
550 CALL CHAR(123,G$)!DESSIN
560 CALL CHAR(122,H$)!DESSIN
570 CALL CHAR(125,I$)!BASES
580 CALL CHAR(126,D$)!ECH.
590 CALL CHAR(124,C$)!BONH.
600 CALL CHAR(127,T$)!TOMBE
610 CALL CHAR(92,X$)!NUAGES
620 CALL CHAR(44,Y$)!EXPLO.
630 CALL CHAR(36,L$)!ALP.
640 CALL CHAR(40,M$)!ALP.
650 CALL HCHAR(23,5,125)
660 CALL HCHAR(5,30,125)
670 CALL SPRITE(#10,140,16,160,26)
680 CALL MAGNIFY(3)
690 !--AFFICHAGE DU DESSIN--
700 CALL HCHAR(5,29,120)
710 CALL HCHAR(6,32,121)
720 CALL HCHAR(7,27,123)
730 CALL HCHAR(10,25,123)
740 CALL HCHAR(12,24,123)
750 CALL HCHAR(7,29,123)
760 CALL HCHAR(5,31,121)
770 CALL HCHAR(10,27,123)
780 CALL HCHAR(15,14,123)
790 CALL HCHAR(19,7,123)
800 CALL HCHAR(21,6,123)
810 CALL HCHAR(24,29,121)
820 CALL HCHAR(22,9,123)
830 CALL HCHAR(22,28,122)
840 CALL HCHAR(21,28,122)
850 CALL HCHAR(16,24,122)
860 CALL HCHAR(23,28,121)
870 CALL HCHAR(24,3,121)
880 CALL HCHAR(17,23,122)
890 CALL HCHAR(20,23,122)
900 CALL HCHAR(22,14,122)
910 CALL HCHAR(23,25,122)
920 CALL HCHAR(22,24,122)
930 CALL HCHAR(13,16,120)
940 CALL HCHAR(20,27,121)
950 CALL HCHAR(19,26,121)
960 CALL HCHAR(18,25,121)
970 CALL HCHAR(17,24,121)
980 CALL HCHAR(14,15,120)
990 CALL HCHAR(15,23,121)
1000 CALL HCHAR(14,22,121)
1010 CALL HCHAR(13,21,121)
1020 CALL HCHAR(12,20,121)
1030 CALL HCHAR(16,14,120)
1040 CALL HCHAR(11,19,121)
1050 CALL HCHAR(17,12,121)
1060 CALL HCHAR(16,11,121)
1070 CALL HCHAR(20,12,121)
1080 CALL HCHAR(17,13,120)
1090 CALL HCHAR(21,13,121)
1100 CALL HCHAR(23,14,121)
1110 CALL HCHAR(24,15,121)
1120 CALL HCHAR(16,22,121)
1130 CALL HCHAR(6,28,120)
1140 CALL HCHAR(18,23,121)
1150 CALL HCHAR(19,22,121)
1160 CALL HCHAR(21,23,121)
1170 CALL HCHAR(22,24,121)
1180 CALL HCHAR(16,10,120)
1190 CALL HCHAR(24,25,121)
1200 CALL HCHAR(17,9,120)
1210 CALL HCHAR(18,8,120)
1220 CALL HCHAR(20,7,120)
1230 CALL HCHAR(8,27,120)
1240 CALL HCHAR(22,6,120)
1250 CALL HCHAR(24,4,120)
1260 CALL HCHAR(24,2,120)
1270 CALL HCHAR(23,9,120)
1280 CALL HCHAR(9,26,120)
1290 CALL HCHAR(21,10,120)
1300 CALL HCHAR(20,11,120)
1310 CALL HCHAR(8,29,120)
1320 CALL HCHAR(9,28,120)
1330 CALL HCHAR(11,25,120)
1340 CALL HCHAR(12,26,120)
1350 CALL HCHAR(14,25,120)
1360 CALL HCHAR(13,24,120)
1370 CALL HCHAR(14,23,120)
1380 CALL HCHAR(11,18,120)
1390 CALL HCHAR(12,17,120)
1400 V=15 :: RANDOMIZE :: GOSUB 1820 :: D
ISPLAY AT(24,3)SIZE(5):"12345"
1410 DISPLAY AT(6,1)SIZE(17)
1420 DISPLAY AT(8,1)SIZE(14)
1430 !--TEST DES NUAGES-----
1440 CALL COINC(ALL,C):: IF C=-1 THEN CAL
```

```
L DELSPRITE(ALL):: GOTO 1400
1450 CALL SPRITE(#10,140,16,160,26)
1460 !---1er TABLEAU--------
1470 CALL KEY(0,K1,S)
1480 CALL COINC(ALL,C):: IF C=-1 THEN CAL
L COLOR(#10,1):: T=1
1490 T=T+1 :: IF T=5+TT THEN CALL COLOR(#
10,16):: T=0
1500 CALL POSITION(#10,X,Y)
1510 IF Y>30 AND Y<70 THEN IF X+Y>173 AND
 X+Y<183 THEN 1920
1520 IF Y<134 AND Y>20 THEN IF X+Y>200 AN
D X+Y<212 THEN 1920
1530 IF Y>134 AND Y<250 THEN IF Y-X>67 AN
D Y-X<77 THEN 1920
1540 IF Y>166 AND Y<220 THEN IF Y+X*5/7>2
37 AND Y+X*5/7<243 THEN 1920
1550 IF Y>250 THEN 1920
1560 IF X>180 AND X<250 THEN 1920
1570 CALL COINC(ALL,C):: IF C=-1 THEN CAL
L COLOR(#10,1):: T=1
1580 IF K1=68 OR K1=69 OR K1=88 OR K1=83
THEN 1680
1590 CALL SOUND(60,-8,15,200,30,200,30,50
00,30)
1600 CALL SOUND(50,110,12)
1610 CALL POSITION(#10,X,Y)
1620 IF X>152 AND X<165 THEN IF Y<32 AND
Y>20 THEN CALL MOTION(#10,0,0):: CALL L
OCATE(#10,160,26):: IF IPO=1 THEN 2650 EL
SE 1770
1630 IF IPO=1 AND X=17 AND Y=225 THEN 177
0
1640 IF IPO=1 THEN 1660
1650 IF X>12 AND X<23 THEN IF Y<232 AND Y
>217 THEN CALL DELSPRITE(#10):: CALL HC
HAR(3,29,140):: CALL HCHAR(3,30,142):: CA
LL HCHAR(4,30,143):: CALL HCHAR(4,29,14
1):: GOTO 2110
1660 CALL MOTION(#10,5,0)
1670 GOTO 1770
1680 IF K1=69 THEN CALL MOTION(#10,-8,0):
: GOTO 1720
1690 IF K1=68 THEN CALL MOTION(#10,0,8)::
 GOTO 1720
1700 IF K1=88 THEN CALL MOTION(#10,8,0)::
 GOTO 1720
1710 IF K1=83 THEN CALL MOTION(#10,0,-8)
1720 CALL SOUND(-50,-5,6,120,10)
1730 IF K1<>83 THEN 1770
1740 FOR I=59 TO 60
1750 CALL PATTERN(#10,I)
1760 NEXT I :: GOTO 1470
1770 FOR I=139 TO 140
1780 CALL PATTERN(#10,I)
1790 NEXT I
1800 GOTO 1470
1810 !----NUAGES-----------
1820 FOR A=2 TO 9
1830 IF RA<>0 THEN 1860
1840 DISPLAY AT(6,1)SIZE(17)BEEP:"evites
les nuages"
1850 DISPLAY AT(8,1)SIZE(14):"et la monta
gne"
1860 CALL SPRITE(#A,92,16,240,(RND*210)+4
0,-8,0)
1870 FOR I=0 TO 60 :: NEXT I
1880 CALL COINC(ALL,C):: IF C=-1 THEN CAL
L DELSPRITE(#A):: GOTO 1860
1890 NEXT A
1900 RETURN
1910 !---EXPLOSION---------
1920 CALL MOTION(#10,0,0)
1930 CALL SOUND(600,-8,0,110,12,110,12,80
0,30)
1940 CALL COLOR(#10,16)
1950 CALL POSITION(#10,X,Y)
1960 CALL SPRITE(#1,44,10,X,Y)
1970 FOR I=1 TO 300 :: NEXT I
1980 CALL LOCATE(#10,160,26):: CALL DELSP
RITE(#1)
1990 CA=CA+1 :: CALL SOUND(200,120,0):: C
ALL HCHAR(24,10-CA,32)
2000 IF IPO=1 THEN AA=AA+1 :: CALL SOUND(
300,150,2):: CALL HCHAR(24,24+AA,127)::
 IPO=0 :: IF AA=3 AND RA<>0 THEN 2820
2010 IF AA=3 THEN 2030
2020 IF CA<>5 THEN 1470
2030 FOR I=1 TO 200 :: NEXT I
2040 !--PLUS D'HELICOPTERE--
2050 DISPLAY AT(10,1)BEEP ERASE ALL:"PREN
EZ DES COURS DE PILOTAGE"
2060 DISPLAY AT(16,1):"   ENCORE UN ESSAI
 OxN"
2070 CALL KEY(0,K,S):: IF S=0 THEN 2070
2080 IF K=79 OR K=111 THEN 2090 ELSE CALL
 CLEAR :: END
2090 TT=0 :: IPO=0 :: RA=0 :: AA=0 :: CA=
0 :: CALL CLEAR :: CALL DELSPRITE(ALL):
: GOTO 650
2100 !---TABLEAU 2---------
2110 CALL SOUND(500,262,2,145,5,392,3)
2120 CALL SOUND(50,262,30)
2130 FOR I=1 TO 3 :: CALL HCHAR(5+I,30,12
6):: CALL SOUND(-80,400*I,11):: NEXT I
2140 GOSUB 2900
2150 CALL SPRITE(#11,128,15,145,229)
2160 IF AA>=1 THEN DISPLAY AT(21,18):"au
suivant" :: GOTO 2180
2170 DISPLAY AT(21,20):"grimpe"
2180 CALL KEY(0,K1,S)
2190 CALL COINC(ALL,C):: IF C=-1 THEN 238
0
2200 CALL POSITION(#11,X,Y)
2210 IF Y>160 AND Y<216 THEN IF Y-X>52 AN
D Y-X<60 THEN 2380
2220 IF Y-X>196 AND Y-X<204 THEN 2380
2230 IF Y>166 AND Y<250 THEN IF Y+X*5/7>2
45 AND Y+X*5/7<251 THEN 2380
2240 IF Y>250 OR X>175 THEN 2380
2250 CALL COINC(ALL,C):: IF C=-1 THEN 238
0
2260 IF K1=68 OR K1=69 OR K1=83 THEN 2300
2270 CALL MOTION(#11,0,0)
2280 FOR I=128 TO 135 :: CALL PATTERN(#11
,I):: NEXT I
2290 GOTO 2180
2300 IF K1=69 THEN CALL MOTION(#11,-8,0):
: GOTO 2340
2310 IF K1=68 THEN CALL MOTION(#11,0,4)::
 GOTO 2340
2320 IF K1=83 THEN CALL MOTION(#11,0,-4):
: GOTO 2340
2330 IF K1=88 THEN CALL MOTION(#11,8,0)
2340 FOR I=36 TO 43 :: CALL PATTERN(#11,I
):: NEXT I
2350 CALL POSITION(#11,X,Y):: IF Y>228 AN
D Y<236 THEN IF X<64 THEN 2530
2360 GOTO 2180
2370 !---CHUTE--------------
2380 CALL MOTION(#11,16,0)
2390 FOR I=1 TO 20 :: CALL SOUND(-200,-8,
0,222,30,222,30,200*I,30)
2400 CALL POSITION(#11,X,Y)
2410 IF X>200 THEN CALL DELSPRITE(#11)::
GOTO 2430
2420 NEXT I
2430 AA=AA+1 :: CALL SOUND(300,150,2):: C
ALL HCHAR(24,25+AA,127)
2440 IF AA<>3 THEN 2140
2450 IF RA<>0 THEN 2820
2460 DISPLAY AT(21,18)
2470 FOR I=1 TO 200 :: NEXT I
2480 DISPLAY AT(10,1)BEEP ERASE ALL:"VOUS
 MANQUEZ D'EXERCICE": : : : :" ENCORE
 UN ESSAI  OxN"
2490 CALL KEY(0,K,S):: IF S=0 THEN 2490
2500 IF K=79 OR K=111 THEN 2510 ELSE END
2510 TT=0 :: RA=0 :: IPO=0 :: CA=0 :: AA=
0 :: CALL CLEAR :: CALL DELSPRITE(ALL):
: GOTO 650
2520 ! --ASCENSION REUSSIE--
2530 CALL LOCATE(#11,48,228)
2540 CALL SOUND(100,2000,2)
2550 CALL SOUND(100,3000,2)
2560 CALL LOCATE(#11,32,228)
2570 CALL SOUND(200,4000,2)
2580 CALL DELSPRITE(#11)
2590 CALL SPRITE(#10,140,16,17,225)
2600 CALL HCHAR(3,29,32,2)
2610 CALL HCHAR(4,29,32,2)
2620 FOR I=1 TO 3 :: CALL HCHAR(5+I,30,32
):: CALL SOUND(-80,400*I,11):: NEXT I
2630 DISPLAY AT(21,18)
2640 IPO=1 :: GOTO 1470
2650 !--MUSIQUE DE REUSSITE-
2660 RESTORE 200
2670 FOR I=1 TO 10
2680 READ F1,D1
2690 F2=F1*3
2700 F3=F1/2
2710 D1=D1*85
2720 CALL SOUND(D1,F1,2,F2,5,F3,5)
2730 NEXT I
2740 RA=RA+1 :: CALL HCHAR(22,6+RA,124)::
 AA=AA+1
2750 IPO=0
2760 IF RA=3 OR AA=3 THEN 2810
2770 FOR A=2 TO 9 :: CALL DELSPRITE(#A)::
 NEXT A :: V=V+3 :: TT=TT+2 :: GOSUB 18
20
2780 DISPLAY AT(6,1)SIZE(17)
2790 DISPLAY AT(8,1)SIZE(14)
2800 GOTO 1470
2810 !----FIN DE JEU--------
2820 FOR I=1 TO 200 :: NEXT I
2830 IF RA=3 AND CA<=2 THEN 2870
2840 DISPLAY AT(5,1)BEEP ERASE ALL:"    M
ISSION EFFECTUEE   ": :"CRASH DE:";CA
;" HELICOPTERE S ": :" MORT DE:";AA-RA;"
ALPINISTE S "
2850 DISPLAY AT(11,1):"  VOUS POUVEZ MIEU
X FAIRE  "
2860 GOTO 2060
2870 DISPLAY AT(7,1)BEEP ERASE ALL:"== VO
US ETES UN CHAMPION ==": :"   VOUS N'AV
EZ CASSE QUE": :"    ";CA;"HELICOPTERE S"
2880 GOTO 2060
2890 !---ATTENTE---------
2900 DISPLAY AT(3,1)SIZE(25):"appuies sur
 ""Q"" et grimpe"
2910 DISPLAY AT(5,1)SIZE(21):"en evitant
les nuages"
2920 CALL KEY(1,K,S):: IF K<>18 THEN 2920
2930 DISPLAY AT(3,1)SIZE(25)
2940 DISPLAY AT(5,1)SIZE(21)
2950 RETURN
2960 !---ECRAN TITRE--------
2970 FOR I=2 TO 11 :: CALL COLOR(I,2,2)::
 NEXT I
2980 CALL MAGNIFY(2)
2990 DISPLAY AT(1,1)ERASE ALL:"SOS HELICO
"
3000 FOR I=1 TO 10
3010 CALL GCHAR(1,2+I,CAR)
3020 XV=SGN(RND-.5)*INT(RND*20)+9 :: YV=S
GN(RND-.5)*INT(RND*20)+9
3030 CALL SPRITE(#I,CAR,2+I,80,110,XV,YV)
3040 RANDOMIZE :: CALL SOUND(80,110*I,5)
3050 NEXT I
3060 CALL DELSPRITE(#4)
3070 FOR I=1 TO 100 :: NEXT I
3080 FOR I=1 TO 3
3090 CALL SOUND(100,1500-100*I,5)
3100 CALL MOTION(#I,0,0)
3110 CALL LOCATE(#I,80-A,78+16*I):: IF A=
8 THEN A=0 :: GOTO 3130
3120 A=8
3130 NEXT I
3140 A=0
3150 FOR I=5 TO 7
3160 CALL SOUND(100,1500-100*I,5)
3170 CALL MOTION(#I,0,0)
3180 CALL LOCATE(#I,112+A,70+2*B)
3190 A=A+10 :: B=B+8 :: NEXT I
3200 FOR I=8 TO 10
3210 CALL SOUND(100,1500-100*I,5)
3220 CALL MOTION(#I,0,0)
3230 CALL LOCATE(#I,132-C,70+2*B)
3240 C=C+10 :: B=B+8 :: NEXT I
3250 A,B,C=0
3260 FOR A=1 TO 10
3270 CALL COLOR(#A,1)
3280 CALL SOUND(100,400,10)
3290 CALL COLOR(#A,A+3)
3300 NEXT A
3310 CALL CLEAR :: FOR I=1 TO 300 :: NEXT
 I
3320 FOR I=2 TO 11 :: CALL COLOR(I,16,2):
: NEXT I :: FOR I=1 TO 10 :: CALL COLOR
(#I,1):: NEXT I
3330 DISPLAY AT(4,1):"LE JEU CONSISTE A R
ECUPERER": :"TROIS ALPINISTES BLOQUES P
AR": :"LE MAUVAIS TEMPS EN ALTITUDE": :"L
E PILOTAGE N'EST PAS FACILE"
3340 DISPLAY AT(12,1):"CAR L'HELICOPTERE
POSSEDE": :"UNE CERTAINE INERTIE DE PLU
S": :"LA MONTAGNE EST PLEINE DE": :"PIEGE
S.UTILISEZ LE CLAVIER"
3350 DISPLAY AT(23,1):"<ENTER>"
3360 CALL KEY(0,K,S):: IF S=0 THEN 3360
3370 CALL CLEAR
3380 FOR I=1 TO 10 :: CALL COLOR(#I,I+2):
: NEXT I
3390 FOR I=1 TO 100 :: NEXT I :: RETURN
```

Le mode édition en Basic TI

Normalement, pour passer en mode "édition", il faut utiliser la commande "EDIT numéro de ligne". Ceci peut être avantageusement remplacé par :

Numéro de ligne et FCTN-E ou FCTN-X (touche fléchées).

Lorsque vous êtes en mode édition, vous pouver corriger la ligne ou passer à une autre ligne en employant les commandes précitées. Pour revenir au mode commande, il suffit d'appuyer sur la touche "ENTER".

Utilisation de la Mini-mémoire

Denise Amrouche

En dehors des utilisations les plus évidentes (sauvegarde rapide de courts programmes en Basic et écriture de programmes en assembleur), la Mini-mémoire apporte d'autres possibilités :

1 - Création de fichiers d'autres types que ceux possibles sur cassettes. Nous montrerons essentiellement comment ces fichiers permettent d'augmenter la taille mémoire du TI-99/4A, des 4Ko de la Mini-mémoire.

2 - Accès à la mémoire centrale de l'ordinateur. Nous donnerons quelques exemples.

3 - Accès à la mémoire VDP (mémoire vive du processeur video) qui gère ce qui concerne l'écran. Nous détaillerons les possibilités ainsi ouvertes, pour accélérer la gestion de l'écran en Basic, et pour créer quelques lutins (sprites).

Fichiers en Mini-mémoire

Rappelons brièvement les différentes caractéristiques d'un fichier, leurs avantages et leurs inconvénients (pour plus de détails, voir le manuel d'utilisation du TI-99/4A, pages 122 et suivantes).

SEQUENTIAL ou RELATIVE

Un fichier ouvert en mode SEQUENTIAL sera lu dans l'ordre successif des différentes fiches. Ce sera le cas de notre premier exemple, où nous avons besoin de lire le fichier une seule fois, d'un bout à l'autre. C'est le seul choix possible sur cassette, et cela impose une lecture assez longue si l'on désire lire seulement la fiche 30 (par exemple), mais cela aura l'avantage de permettre un fichier de type VARIABLE en Mini-mémoire, et de gagner ainsi de la place.

A l'opposé, un fichier ouvert en mode RELATIVE permet, grâce à la clause REC, l'accès direct à une fiche de numéro donné. Ce sera le cas de notre second exemple. Un tel fichier ne pourra être ouvert que FIXED (longueur fixe). Il faudra donc, dans la mesure du possible, fixer cette longueur de façon à gagner de la place mémoire.

INPUT, OUTPUT, UPDATE et APPEND

En Mini-mémoire, nous utiliserons UPDATE, qui permet indifféremment d'écrire ou de lire dans le fichier.

DISPLAY ou INTERNAL

En type DISPLAY, tout se passe comme sur l'écran. C'est-à-dire, par exemple, que 12 est stocké comme la suite des codes ASCII 32, 49, 50 et 32, tout nombre sur l'écran étant précédé et suivi d'un espace. Si on demande "PRINT #1:12,34", le fichier notera 1 espace, 1, 2, des espaces jusqu'à la moitié d'une ligne d'écran, 1 espace, 3, 4, et 1 espace, ce qui prendra énormément de place. Ceci explique l'erreur commise page 15 du mode d'emploi jaune de la Mini-mémoire. A la ligne "PRINT #3", il faut mettre "PRINT #3:A:B:C:D", pour séparer les quatre enregistrements.

Au contraire, en type INTERNAL, les caractères sont codés comme dans la machine. Une chaîne A$ occupe LEN(A$)+1 octets (l'octet supplémentaire servant à noter la longueur de la chaîne).

Un nombre A occupe toujours 9 octets : 1 pour sa longueur et 8 pour son codage en virgule flottante. On remarque donc qu'un nombre, même entier ou petit, prend toujours 9 octets alors que le nombre 123 mis en chaîne ne prendra plus que 4 octets. Cette remarque sera utilisée dans l'exemple 1.

VARIABLE ou FIXED

Un fichier SEQUENTIAL, pourra être en VARIABLE.

Pour un fichier RELATIVE, FIXED est imposé et, pour les raisons déjà citées, il sera bon de préciser cette longueur fixe.

Exemple 1 : Dessin d'une voiture de course

En Basic simple, la Mini-mémoire ne peut pas être utilisée pour ajouter des lignes de Basic à un programme qui remplirait déjà la console, mais vous pouvez utiliser la Mini-mémoire en ouvrant un fichier pour y stocker des DATA qui prennent beaucoup de place dans le programme.

L'exemple 1 est un programme qui doit dessiner sur l'écran une belle voiture de course, qui pourra vous servir d'introduction à un programme de course automobile. Il y a beaucoup de caractères à redéfinir (CALL CHAR), et ensuite beaucoup de caractères à mettre en place (CALL HCHAR).

Notre démarche sera la suivante :

1 - insérer la Mini-mémoire (sauvegarder son contenu, s'il y a lieu);

2 - taper le programme FICHFOR, dans lequel se trouvent tous les DATA qui seront lus et transférés sur un fichier en Mini-mémoire;

3 - faire RUN, ce qui effectue le transfert;

4 - sauvegarder le programme sur cassette ou disquette;

5 - taper le programme FORMULE1, qui lira le fichier et fera le dessin sur l'écran;

6 - faire RUN;

7 - sauvegarder le programme sur disquette ou cassette.

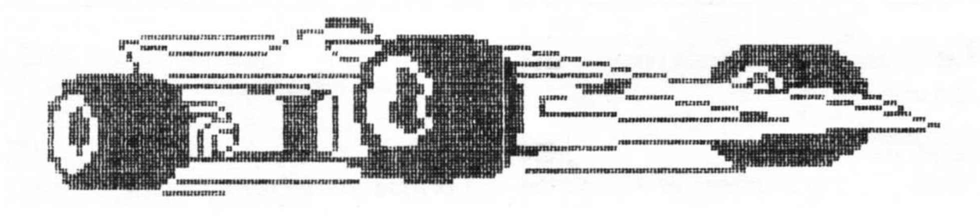
Si le dessin est joli et s'il n'y a pas d'erreur, sauver le fichier Mini-mémoire sur cassette en utilisant EASY BUG S7000 TO? 8000.

S'il y a une erreur de graphisme, bien vérifier le programme FORMULE1. Si cela ne va toujours pas, recharger FICHFOR, le vérifier et recommencer la procédure en 2.

Pour une utilisation ultérieure, il vous suffira de charger le fichier en Mini-mémoire (EASY BUG option L), puis le programme FORMULE1 en Basic. Vous avez un programme Basic très court, pour un dessin assez compliqué, et il vous reste de la place pour faire un beau programme.

Commentaires sur le programme FICHFOR

```
100 REM ****************
105 REM *  Programme    *
110 REM *   Fichfor     *
115 REM ****************
120 REM
125 REM Data pour les
130 REM CALL CHAR
135 REM
180 DATA 0000000000000007,000000
00000000FF,0000000001011E1,00000
000F8FC0C02
```



```
190 DATA 000000000000007F,000000
00000000F,000000000000F3F,38271
0100808FEFF
200 DATA 00807F0000007F8,0001FE0
00000FF,41810000000FF,FF0183770
E0CFC0C
210 DATA FFFFFF1F0F6F7773,FCFFFF
FFFFFFFFFF,00F00F80FFC0FFC,00000
0E01E01FC03
220 DATA 0000000000C03EC1,000000
01030707FF,007FFFFFFFC79B3D,00FC
FEFFFFFFFFFF
230 DATA 0000000080C0C0E0,3F3F67
C7CB9B9D9D,FFFFFFFFFFFFFFFF,FFE0
E0FFF0FFF9F2
240 DATA FF0F00008040C02,F807FE7
C7C7C7C7C,1C189898989888C,7B7B7
7F7F7F7FB7B
250 DATA E0E0E7E4E4E7E0E,3F1F800
000FF,C0807F000000FF,000000F8070
0FF
260 DATA E51F00008078FF,FF03F807
,E0E038867907,000000000806018
270 DATA 9D9D9D898BC3476F,F4F5F5
F5F5FFF0C,E0202040BFFF,7C7C7C7CF
FFF
280 DATA 8C8E8E8707FF3F0F,370F0F
FFFFFFFFFF,E0C0C0FFF880801F,0000
00FF000000FF
290 DATA 0000FF00000000FF,0000FF
07030100FF,000FFFFFFFFFFFC,07FFF
FFFPFFEFC
300 DATA FFE0E0C08,E,3F3F1F,FFFF
FF,C0BF00FF,00FF008,00FF,0FF,FF3
F,FE8
310 REM
315 REM Data pour
320 REM positionner les
325 REM caracteres sur
330 REM l'ecran
335 REM
340 DATA 1210102,1211103,1212103
,1213104,1214105,1215106,1216107
350 DATA 1309108,1310109,1311110
,1312111,1313112,1314113,1315114
,1316115
360 DATA 1317116,1318117,1319118
,1320119,1321120,1322121,1323122
370 DATA 1409123,1410124,1411125
,1412126,1413127,1414128,1415129
,1416124
380 DATA 1417130,1418131,1419132
,1420133,1421134,1422135,1423136
,1424137
390 DATA 1509138,1510124,1511139
,1512140,1513141,1514142,1515143
,1516124
400 DATA 1517144,1518145,1519146
,1520147,1521148,1522149,1523150
,1524151
410 DATA 1609152,1610153,1611154
,1612155,1613156,1614157,1615158
,1616159
415 REM
420 REM Lecture des Data
425 REM et mise en
430 REM fichier
435 REM
450 CALL INIT
460 OPEN #1:"MINIMEM",SEQUENTIAL
,VARIABLE,UPDATE,INTERNAL
470 FOR I=102 TO 159
480 READ A$
490 PRINT #1:A$
500 NEXT I
510 FOR I=1 TO 62
520 READ B
530 PRINT #1:B
540 NEXT I
550 CLOSE #1
560 STOP
```

Le programme utilise les caractères 102 à 159; les lignes 180 à 300 donnent les patrons de chacun de ces caractères.
Pour placer ces caractères, nous aurons besoin de la ligne C et du No CH du caractère. Pour éviter de stocker trois nombres par caractère, ce qui prendrait 27 octets, nous utiliserons un seul nombre, codé de cette façon :

- ligne : 2 chiffres;
- colonne : 2 chiffres;
- caractère : 3 chiffres.

Ainsi, 1210102 correspond à la ligne 12, la colonne 10, et le caractère 102.

Les lignes 450 à 550, après avoir réinitialisé la Mini-mémoire, lisent les DATA et les mettent dans un fichier.

Commentaires sur le programme FORMULE1

```
100 REM ****************
105 REM *  Programme    *
110 REM *   Formule1    *
115 REM ****************
120 REM
125 REM Dessin en
130 REM utilisant le
135 REM fichier de
140 REM donnees
145 REM
160 CALL CLEAR
170 OPEN #1:"MINIMEM",SEQUENTIAL
,UPDATE,VARIABLE,INTERNAL
180 FOR I=102 TO 159
190 INPUT #1:A$
200 CALL CHAR(I,A$)
210 NEXT I
220 FOR I=1 TO 62
230 INPUT #1:B
240 L=INT(B/100000)
250 U=B-L*100000
260 C=INT(U/1000)
270 CH=U-C*1000
280 CALL HCHAR(L,C,CH)
290 NEXT I
300 CLOSE #1
310 GOTO 310
```

La démarche est inverse, le fichier est lu et utilisé d'abord pour définir les caractères (180 à 210), puis pour les placer (220 à 290). On doit alors décoder notre nombre de sept chiffres, pour en extraire L, C et CH. On a évité les fonctions STR$ et SEG$ pour des raisons de rapidité d'exécution.

Exemple 2 : Airs de chansons enfantines

La musique est un autre exemple utilisant beaucoup de DATA, donc beaucoup de place. Dans le même esprit que dans le premier exemple, la procédure sera la suivante :

1 - insérer la Mini-mémoire (sauvegarder son contenu s'il y a lieu);
2 - taper le programme FICHMUS;
3 - faire RUN;
4 - le sauvegarder sur cassette ou disquette;
5 - taper le programme musique et le tester.

Si le résultat est bon, sauvegarder le programme MUSIQUE, puis sauvegarder le fichier Mini-mémoire (EASY BUG S 7000 TO? 8000).
Si le résultat n'est pas au point, vérifier MUSIQUE, le sauvegarder sur cassette, puis recharger FICHMUS et le vérifier. Refaire RUN, sauvegarder FICHMUS, charger MUSIQUE. Tout devrait être bon. Pour une utilisation ultérieure : charger la Mini-mémoire (EASY BUG option L), puis le programme MUSIQUE; ensuite, faire RUN.

Commentaires sur le programme FICHMUS

```
100 REM ****************
105 REM *  Programme    *
110 REM *   Fichmus     *
115 REM ****************
120 REM
125 REM Donnees pour
130 REM "Au clair de
135 REM la lune"
140 REM
145 REM Fiches 0 a 11
150 REM
170 DATA 1,392,196,196,1,392,196
,196,1,392,294,247,1,440,294,247
```

```
,2,494,196,196,2
,440,294,262,1,392,196,196
180 DATA 1,494,196,196,1,440,294
,262,1,440,294,262,3,392,196,196
,1,40000,40000,4
0000
185 REM
190 REM Fiches 12 a 35
195 REM
200 DATA 1,440,139,139,1,440,139
,139,1,440,220,196,1,440,220,196
,2,330,139,139,2
,330,220,196
210 DATA 1,440,139,139,1,392,139
,139,1,370,220,196,1,330,220,196
,3,294,147,147,1
,40000,40000,40000
220 DATA 1,392,196,196,1,392,196
,196,1,392,294,247,1,440,294,247
,2,494,196,196,2
,440,294,262
230 DATA 1,392,196,196,1,494,196
,196,1,440,294,262,1,440,294,262
,3,392,294,247,1
,40000,40000,40000
235 REM
240 REM "Meunier, tu
242 REM dors"
245 REM
250 REM Fiches 36 a 77
255 REM
280 DATA 2,440,40000,40000,4,440
,175,175,2,523,262,220,4,349,131
,131,1,330,262,2
20,1,349,262,220
290 DATA 4,392,165,165,1,349,262
,233,1,392,262,233,2,440,175,175
,2,349,262,220,2
,440,262,220
300 DATA 4,440,175,175,2,523,262
,220,4,349,131,131,1,330,262,220
,1,349,262,220,4
,392,165,165,1,349,262,233
310 DATA 1,392,262,233,4,349,262
,220,6,40000,40000,40000,1,440,4
0000,40000
320 DATA 1,440,40000,40000,2,440
,175,40000,1,440,262,220,1,440,2
62,220,2,440,131
,40000,1,440,262,220
330 DATA 1,440,262,220,4,523,175
,40000,2,440,131,40000,1,440,262
,220,1,440,262,2
20,2,392,165,40000
340 DATA 1,392,262,233,1,392,262
,233,2,523,131,40000,1,523,262,2
33,1,523,262,233
,4,440,175,40000
350 DATA 2,40000,131,40000
355 REM
360 REM "Frere Jacques"
365 REM
370 REM Fiches 78 a 81
375 REM
400 DATA 2,392,247,247,2,440,262
,262,2,494,294,294,2,392,247,247
405 REM
410 REM Fiches 82 a 84
415 REM
420 DATA 2,494,196,196,2,523,220
,220,4,587,247,196
425 REM
430 REM Fiches 85 a 90
435 REM
440 DATA 1,587,247,247,1,659,247
,247,1,587,262,262,1,523,262,262
,2,494,294,294,2
,392,247,247
445 REM
450 REM Fiches 91 a 93
455 REM
460 DATA 2,392,247,247,2,294,196
,196,4,392,247,247
470 REM
480 REM Mise sur fichier
490 REM
500 CALL INIT
510 OPEN #1:"MINIMEM",RELATIVE,F
IXED 36,INTERNAL,UPDATE
520 FOR I=1 TO 94
530 READ A,B,C,D
540 PRINT #1:A,B,C,D
550 NEXT I
560 CLOSE #1
570 STOP
```

Nous ouvrirons cette fois un fichier RELATIVE, donc avec possibilité d'accès direct à une fiche donnée, d'une part parce que trois airs sont proposés, d'autre part parce qu'un même air peut reprendre plusieurs fois la même ligne de partition.

Chaque fiche (ligne 540) enregistrera les renseignements concernant une note :

- coefficient de durée;
- note;
- première note d'accompagnement;
- deuxième note d'accompagnement.

Chaque fiche aura donc une longueur de 36 octets (9⋆4).

Pour garder une vitesse d'exécution de la musique à peu près satisfaisante, on a renoncé à tout codage ou traduction en chaîne des notes. Il n'était pas question non plus de lire toutes les notes avant exécution, ce qui aurait lassé l'auditeur impatient, et occupé de la place en mémoire, alors que notre but est d'en économiser dans le programme principal.

Commentaires sur le programme MUSIQUE

```
100 REM ***************
105 REM *  Programme    *
110 REM *   Musique     *
115 REM ***************
120 REM
125 REM Musique a partir
130 REM d'un fichier en
135 REM Mini-memoire
140 REM
150 OPEN #1:"MINIMEM",RELATIVE,F
IXED 36,INTERNAL,UPDATE
160 CALL CLEAR
170 PRINT : : : :"CHOISISSEZ:":
:
180 PRINT "1  AU CLAIR DE LA LUN
E": :
190 PRINT "2  MEUNIER,TU DORS":
:
200 PRINT "3  FRERE JACQUES": :
:
210 CALL KEY(0,K,S)
220 IF (K<1)+(K<49)+(K>51)<>0 TH
EN 210
230 ON K-48 GOTO 260,400,510
240 REM
245 REM Au clair de la
250 REM lune
255 REM
260 L=250
270 FOR J=1 TO 2
280 FOR I=0 TO 11
290 GOSUB 780
300 NEXT I
310 NEXT J
320 FOR I=12 TO 35
330 GOSUB 780
340 NEXT I
350 GOTO 210
360 REM
370 REM Meunier, tu dors
380 REM
400 L=200
410 FOR I=36 TO 77
420 GOSUB 780
430 NEXT I
440 FOR I=58 TO 77
450 GOSUB 780
460 NEXT I
470 GOTO 210
480 REM
490 REM Frere Jacques
500 REM
510 L=300
520 FOR K=1 TO 2
530 FOR J=1 TO 2
540 FOR I=78 TO 81
550 GOSUB 780
```

```
560 NEXT I
570 NEXT J
580 FOR J=1 TO 2
590 FOR I=82 TO 84
600 GOSUB 780
610 NEXT I
620 NEXT J
630 FOR J=1 TO 2
640 FOR I=85 TO 90
650 GOSUB 780
660 NEXT I
670 NEXT J
680 FOR J=1 TO 2
690 FOR I=91 TO 93
700 GOSUB 780
710 NEXT I
720 NEXT J
730 NEXT K
740 GOTO 210
750 REM
755 REM Sous-programme
760 REM de lecture et de
765 REM jeu des notes
770 REM
780 INPUT #1,REC I:A,B,C,D
790 CALL SOUND(L*A,B,1,C,7,D,10)
800 RETURN
```

Pour chaque air, on utilise la clause REC, avec la variable I qui indique la fiche que l'on désire lire. On peut ainsi relire des fiches... Le programme comporte de nombreuses lignes courtes et occupe peu de place en mémoire.

Accès à la mémoire centrale

La Mini-mémoire étant insérée, vous disposez des deux routines : CALL PEEK et CALL LOAD.

CALL PEEK(adresse décimale, variable X, variable Y, ...) lit le contenu de l'octet situé à l'adresse indiquée, et le met en valeur décimale dans X, lit le contenu de l'adresse suivante et le met dans Y, ... Plusieurs adresses peuvent être lues, séparées par "".

CALL LOAD(adresse décimale, X, Y, ...) a l'effet inverse. Il met dans l'adresse indiquée l'octet correspondant à la valeur décimale X, et dans l'adresse suivante l'octet correspondant à la valeur décimale Y, ...

Quelques adresses intéressantes

A) −31888 et −31887.

Dans ces deux adresses consécutives (>8370 et >8371 en hexa) se trouve la dernière adresse VDP utilisable.
Faites en Basic CALL PEEK(−31888,X,Y) puis PRINT X,Y.

1) Si vous n'avez pas de contrôleur de disquettes, vous lisez X=63, Y=255. C'est-à-dire que dans l'adresse −31888 (>8370) se trouve la valeur >3F, et dans −31887 (>8371) la valeur >FF.
La plus haute adresse utilisable en mémoire vive du processeur vidéo est donc >3FFF ou, en décimal, 63*256+255=16383, ce qui est effectivement la plus haute adresse de la MEV VDP (cf. mode d'emploi jaune de la Mini-mémoire, pages 75 et 76).

2) Si vous avez un contrôleur de disquettes, vous obtenez X=55 et Y=215. La plus haute adresse disponible dans la MEV VDP est donc 55*216+215=14295 ou >37D7.
Si, avec un contrôleur de disquettes, vous tapez dans l'ordre :
CALL FILES(1)
NEW
CALL PEEK(−31888,X,Y)
PRINT X,Y
vous obtenez X=59 et Y=227. La plus haute adresse utilisable est donc 59*256+227=15331.

N.B. En Basic simple, le programme est stocké en MEV VDP, il est donc intéressant d'y avoir un maximum de place disponible, d'où l'intérêt si vous possédez un lecteur de disquettes, de taper CALL FILES(1) et NEW avant de taper ou de charger un long programme Basic, ou un programme utilisant de grands ou de nombreux tableaux. Vous récupérez ainsi 15331−14295=1036 octets. Il vous manque encore 16383−15331=1052 octets, par rapport à ceux qui utilisent les cassettes.

B) −31878 (>837A).

C'est dans cette adresse que l'on doit indiquer à l'ordinateur le nombre de lutins (sprites) en mouvement.
Si nous mettons en mouvement trois lutins, nous ferons :
CALL LOAD(−31878,3)

C) −31788 (>83D4).

Une copie du contenu de cette adresse est mise dans le registre 1 du processeur vidéo, chaque fois qu'une touche est pressée alors que l'ordinateur est en mode de scrutation, c'est-à-dire qu'il attend la frappe d'une touche. Or, le registre 1 est celui qui détient des renseignements sur le mode d'affichage et sur la taille des lutins. Nous pouvons donc, en utilisant l'adresse −31788, agir sur ces facteurs.

En détail, en numérotant les bits de 0 à 7, et de gauche à droite :

Bit 0 : 1 toujours
Bit 1 : 1 toujours
Bit 2 : 1 en Basic
Bit 3 : 1 = affichage en mode texte, 0 = autre affichage
Bit 4 : 1 = affichage en mode multicolore, 0 = autre affichage
Bit 5 : 0 toujours
Bit 6 : 0 = lutins définis par un seul caractère, 1 = lutins définis par quatre caractères
Bit 7 : 0 = lutins de dimension normale, 1 = lutins agrandis

Montrons un peu ce qu'est le mode multicolore.

- Remplissez votre écran en tapant n'importe quoi.
- Tapez CALL LOAD(−31788,232).
- Faites ENTER.
- Appuyez sur n'importe quelle touche.
  L'écran est à présent coloré, chaque caractère s'étant divisé en quatre carrés de couleur. C'est le mode multicolore.
  232=>E8=11101000 (le bit 4 est à 1).

Pour revenir à l'état normal :

- Tapez en aveugle CALL LOAD(−31788,224) (224=>E0=11100000), ENTER puis appuyez sur une touche;
- Ou éteignez.

Par défaut, le mode texte est transparent sur fond cyan; il n'est donc pas possible, sans un petit programme en assembleur, de visualiser ce mode d'affichage.
Nous vous avons présenté les adresses les plus utiles en Basic. Il y en a bien sûr d'autres ayant un usage bien particulier pour l'écriture de programmes en assembleur.

Accès à la mémoire vive du processeur vidéo

La mémoire du processeur vidéo est celle qui gère l'écran. En plus, elle contient le programme. En Basic Etendu, si vous disposez d'une extension de mémoire, une partie du programme y est stocké. Vous disposez, la Mini-mémoire étant insérée, de deux instructions d'accès à la MEV VDP.

CALL PEEKV(adresse décimale,X,Y,...) lit à l'adresse indiquée et met le premier octet lu dans X en base 10, le second dans Y en base 10... Des adresses non consécutives peuvent être explorées en séparant par "".
CALL POKEV(adresse décimale,X,Y,...) a l'effet inverse. Il met à partir de l'adresse indiquée les valeurs décimales X, Y,...

Nous allons donner de nombreux exemples d'utilisation. Signalons que les adresses données sont celles utilisées par le Basic (en assembleur, la disposition des différentes tables est autre, et dépend du mode graphique

utilisé); en outre, dans cette MEV VDP, les codes ASCII sont décalés de 96. C'est-à-dire que, par exemple, l'espace (code ASCII 32) est codé 32+96 soit 128. Nous dirons que 128 est le code ASCII interne pour le caractère "espace".

Table d'écran

Cette table, qui s'étend des adresses 0 à 767 (>0000 à >02FF), accorde une adresse à chaque emplacement de l'écran, dans laquelle elle met le code ASCII interne du caractère occupant cet emplacement. Les adresses 0 à 31 correspondent aux 32 colonnes de la première ligne, les adresses 32 à 63 aux 32 colonnes de la seconde ligne, etc...

Exemple : votre écran étant vide, tapez :
CALL PEEKV(4,X)
PRINT X
Vous obtenez X=128 car, à l'emplacement d'adresse 4 (donc ligne 1 - colonne 5), il y a un espace (code ASCII interne 128).

Autre exemple : tapez CALL POKEV(47,161)
Vous obtenez un A sur la ligne 2 - colonne 16, car le code ASCII interne de A est 65+96 soit 161.

Une règle : pour afficher le caractère de code ASCII H à la ligne L (1<=L<=24), colonne C (1<=C<=32), il faut mettre H+96 à l'adresse 32*L+C−33, ce qui se fait par CALL POKEV(32*L+C−33,H+96).

L'instruction habituelle CALL HCHAR(L,C,H,2) s'écrira CALL PEEKV(32*L+C−33,H+96,H+96).

L'instruction CALL GCHAR(L,C,X) s'écrira CALL PEEKV(32*L+C−33,X), où X est égal à X−96.

Cette traduction présente dans certains cas deux avantages :

1 - gain de place en mémoire;
Exemple : pour dessiner un échiquier ou un damier, vous avez à mettre en place un dessin sur un carré de deux caractères par deux. Programme habituel :
CALL HCHAR(X,Y,A)
CALL HCHAR(X,Y+1,B)
CALL HCHAR(X+1,Y,C)
CALL HCHAR(X+1,Y+1,D)
Ceci peut être remplacé par :
CALL POKEV(32*X+Y−33,A+96,B+96,"",32*X+Y−1,C+96,D+96)

2 - rapidité d'exécution

```
100 REM ****************
101 REM *Affichage      *
102 REM *d'une chaine   *
103 REM *en un endroit  *
104 REM *donne          *
105 REM ****************
106 REM
110 CALL CLEAR
120 A$="1004BONJOUR,COMMENT ALLE
Z-VOUS?"
130 GOSUB 5000
200 GOTO 200
5000 REM
5002 REM Sous-programme
5004 REM d'affichage
5006 REM
5010 L=VAL(SEG$(A$,1,2))
5020 C=VAL(SEG$(A$,3,2))
5030 LO=LEN(A$)-5
5040 FOR I=0 TO LO
5050 CALL HCHAR(L,C+I,ASC(SEG$(A
$,5+I,1)))
5060 NEXT I
5070 RETURN
```

Dans ce programme Basic simple et classique, on utilise un sous-programme qui lit la ligne d'affichage, la colonne et le texte. A l'exécution, on voit le message s'afficher peu à peu.
Autre solution :

```
100 REM ****************
101 REM *Affichage      *
102 REM *d'une chaine   *
103 REM *par la         *
104 REM *fonction       *
105 REM *CALL POKEV     *
106 REM ****************
107 REM
110 CALL CLEAR
120 CALL POKEV(291,162,175,174,1
70,175,181,178,129,163,175,173,1
73,165,174,180,1
40,161,172,172,165,186,141,182)
130 CALL POKEV(314,175,181,179,1
59)
200 GOTO 200
```

291=32*10+4−33
162=66+96 (66=code ASCII de B)
Ici, le message s'affiche instantanément.

Table des couleurs

Chaque groupe de caractères se voit attribuer une adresse dans cette table, dans laquelle seront codées la couleur du caractère et la couleur du fond.
En assembleur, les codes des couleurs sont diminués de 1 par rapport à leur code en Basic; ainsi, le transparent devient 0, le noir devient 1, etc...
Pour avoir un groupe de caractères en bleu foncé sur fond vert vert, il faut mettre la valeur >43 dans l'adresse associée à la couleur de ce groupe.

Une règle : pour colorer un groupe donné en caractère de couleur C (codage Basic 1<=C<=16) sur fond de couleur F (1<=F<=16), il faut mettre 16*C+F−17 dans l'adresse associée à ce groupe.

Adresse : 783 (>30F) - gpe 0

Exemple : tapez CALL POKEV(788,11,11,11).
Quand vous faites ENTER, certains caractères de votre texte deviennent provisoirement jaunes, l'effet est temporaire. Exemple : dans un programme, CALL PEEKV à une des adresses indiquées peut vous permettre de connaître la couleur de tel ou tel groupe et d'agir en conséquence. C'est une instruction qui n'existe pas en Basic.

Exemple : vous avez un long texte à afficher sur l'écran et vous ne voulez pas fatiguer le joueur ou lecteur par l'effet de "scroll" (déroulement); vous mettez pour cela tous les caractères en transparent, vous affichez votre texte, puis vous rétablissez les couleurs. Il y a alors un certain effet de papillotement car les couleurs se rétablissent groupe par groupe.

```
100 REM ****************
105 REM *Affichage d'un*
110 REM *texte d'un     *
112 REM *seul coup      *
114 REM ****************
116 REM
120 CALL CLEAR
130 FOR I=1 TO 12
140 CALL COLOR(I,1,1)
150 NEXT I
160 PRINT "ESSAI D'AFFICHAGE D'U
N LONG MESSAGE": : : :
170 PRINT "La quatrieme planete
etait  celle du businessman.Cet
   homme etait s
i occupe qu'il"
180 PRINT "ne leva meme pas la t
ete a  l'arrivee du petit prince
.   _Bonjour,lui
 dit celui-ci."
190 PRINT "Votre cigarette est e
teinte. _Trois et deux font cinq
. Cinq et sept
douze.Douze et"
200 PRINT "trois quinze.Bonjour.
"

250 FOR I=1 TO 12
260 CALL COLOR(I,2,1)
270 NEXT I
300 GOTO 300
```

Remplacez les lignes 130,140 et 150 par l'unique ligne :

130 CALL POKEV(784,0,0,0,0,0,0,0,0,0,0,0,0) et les lignes 250, 260 et 270 par l'unique ligne :
290 CALL POKEV(784,16,16,16,16,16,16,16,16,16,16,16,16)

```
100 REM ****************
102 REM *Affichage d'un*
104 REM *texte d'un    *
106 REM *seul coup     *
108 REM ****************
110 REM
120 CALL CLEAR
130 CALL POKEV(784,0,0,0,0,0,0,0
,0,0,0,0,0)
160 PRINT "ESSAI D'AFFICHAGE D'U
N LONG MESSAGE": : : :
170 PRINT "La quatrieme planete
etait  celle du businessman.Cet
   homme etait s
i occupe qu'il"
180 PRINT "ne leva meme pas la t
ete a  l'arrivee du petit prince
.  _Bonjour,lui
 dit celui-ci."
190 PRINT "Votre cigarette est e
teinte. _Trois et deux font cinq
.  Cinq et sept
douze.Douze et"
200 PRINT "trois quinze.Bonjour.
"
290 CALL POKEV(784,16,16,16,16,1
6,16,16,16,16,16,16,16)
300 GOTO 300
```

Observez la différence. L'exécution étant plus rapide, il n'y a plus de papillotement.

Table des patrons (descripteurs)

Le caractère ASCII "J" (code ASCII interne J+96) est décrit des adresses 8*(J+96) à 8*(J+96)+7 inclus. Votre patron est une chaîne de 16 caractères alphanumériques (0 à 9, A à F). Vous le coupez en 8 tranches de 2 et, pour utiliser l'instruction CALL POKEV, vous traduisez chaque octet en décimal.
>3A devient 3*16+10=58.

Exemple : le curseur vous gêne ou sa forme ne vous plaît pas, modifiez-le. Le curseur a 30 comme code ASCII; il est donc décrit à partir de l'adresse 8*(30+96)=1008.
Pour le faire disparaître :
CALL POKEV(1008,0,0,0,0,0,0,0,0)
Pour lui donner la forme d'un losange :
CALL POKEV(1008,24,60,126,255,255,126,60,24)
Le patron du losange est "183C7EFFFF7E3C18"
Pour qu'il soit plus gros :
CALL POKEV(1008,255,255,255,255,255,255,255,255)
Cette transformation reste valable jusqu'à ce que vous éteigniez la console ou que vous rétablissiez le curseur habituel par :
CALL POKEV(1008,0,124,124,124,124.124 124,124)

Lutins (sprites)

La gestion des lutins nécessite deux tables :
- la table des attributs, qui garde en mémoire les caractères qui les définissent, leurs couleurs et leurs positions à chaque instant;
- la table des vitesses qui garde en mémoire leurs vitesses horizontales et verticales.

La table des attributs des lutins commence à l'adresse 768 (>300). Chaque lutin occupe quatre adresses consécutives et il faut mettre 208 (>D0) à la fin de cette table pour indiquer qu'il n'y a pas d'autres lutins. Comme la table des couleurs commence à l'adresse 783, si nous voulons simultanément du texte ou du graphisme utilisant des codes ASCII bas, nous ne pouvons utiliser que 3 lutins. Si nous mettons uniquement des lutins, nous pourrons en mettre plus.

Pour chaque lutin, nous devons indiquer successivement la position verticale, la position horizontale, le code ASCII et la couleur.

Position verticale du lutin

255 pour la première ligne (ligne de points), 0 pour la seconde ligne, 1 pour la troisième ligne, et ainsi de suite jusqu'à 190 pour la dernière ligne. Chaque ligne de caractères étant divisée en huit lignes de points, il y a 24*8=192 positions verticales possibles.
Les positions 191 à 254 (sauf 208) sont autorisées, mais le lutin est alors invisible.

Position horizontale du lutin

0 pour la première colonne (colonne de points), 1 pour la deuxième, et ainsi de suite jusqu'à 255 pour la dernière colonne.

Code ASCII

Il s'agit du code interne du caractère dont le patron est utilisé pour la définition du lutin.

Couleur du lutin

Même couleur qu'en Basic moins un (−1).

```
100 REM ****************
101 REM *Exemples de   *
102 REM *sprites en    *
103 REM *Basic simple  *
104 REM *grace a la    *
105 REM *Mini-memoire  *
106 REM ****************
107 REM
110 CALL CLEAR
120 CALL POKEV(768,80,170,180,13
)
130 CALL POKEV(772,100,45,169,4)
140 CALL POKEV(776,45,67,153,12)
150 CALL POKEV(780,78,45,153,10,
208)
160 CALL LOAD(-31788,224)
170 GOSUB 250
180 CALL LOAD(-31788,225)
190 GOSUB 250
200 CALL LOAD(-31788,226)
210 GOSUB 250
220 CALL LOAD(-31788,227)
230 GOSUB 250
240 GOTO 160
250 CALL KEY(0,K,S)
260 IF S<1 THEN 250
270 RETURN
```

Le programme crée quatre lutins utilisant les caractères prédéfinis par l'ordinateur. En appuyant sur n'importe quelle touche, vous changez la taille des lutins en utilisant ce qui a été dit sur l'adresse −31788.
224=>E0 : petit lutin sur un caractère
225=>E1 : grand lutin sur un caractère
226=>E2 : petit lutin sur quatre caractères
227=>E3 : grand lutin sur quatre caractères
La table de vitesse des lutins commence à l'adresse >780 (1920) et se superpose à la table des patrons. 1920 est le début de la description du caracère ASCII 144, ce qui nous explique qu'en Basic Etendu 143 soit le dernier code ASCII disponible pour définir un caractère. Chaque lutin occupe quatre octets dans la table.

1er octet : vitesse verticale (0 à 255).
De 0 à 127 : vitesse positive.
De −128 à −1 : vitesse négative. Augmentée de 256, donne un entier de 128 à 255.

2ème octet : vitesse horizontale (même convention que pour la vitesse verticale).

3ème et 4ème octets : ces adresses sont utilisées par la machine (routine d'interruption); il est donc inutile de les gérer.

Exemple :

5 CALL CLEAR
10 CALL POKEV(768,80,170,180,13,208)
20 CALL POKEV(1920,100,200)
30 CALL LOAD(−31878,1)
40 GOTO 40

crée un lutin T se déplaçant de haut en bas (vitesse verticale positive) et de droite à gauche (vitesse horizontale négative : 200−256=−56). Après l'arrêt, tapez CALL LOAD(−31878,0) pour faire disparaître tout lutin.

```
100 REM ***************
101 REM *  Exemple de  *
102 REM *  sprites en  *
103 REM * Basic simple *
104 REM *    avec la   *
105 REM * Mini-memoire *
106 REM ***************
107 REM
110 CALL CLEAR
120 CALL LOAD(-31788,227)
130 PRINT "APPUYEZ SUR UNE TOUCH
E"
140 CALL KEY(0,K,S)
150 IF K<1 THEN 140
160 CALL CLEAR
170 PRINT "D POUR ACCELERER"
180 PRINT "S POUR RALENTIR"
190 CALL CHAR(33,"7071FFE3E3FF71
7")
200 CALL CHAR(34,"")
210 CALL CHAR(35,"0080E0E0E0E08"
)
220 CALL POKEV(768,34,2,128,4,20
8)
230 CALL LOAD(-31878,1)
240 CALL POKEV(1920,0,0)
250 CALL KEY(0,K,S)
260 IF S=0 THEN 250
270 CALL PEEKV(1921,V)
280 IF K=83 THEN 350
290 IF K<>68 THEN 250
300 IF V>126 THEN 330
310 V=V+1
320 GOTO 340
330 V=127
340 GOTO 390
350 IF V<1 THEN 380
360 V=V-1
370 GOTO 390
380 V=0
390 CALL POKEV(1921,V)
400 V1=INT(V/100)
410 V2=V-100*INT(V/100)
420 V3=INT(V2/10)
430 V4=V2-10*V3
440 CALL POKEV(496,144+V1,144+V3
,144+V4)
450 GOTO 250
```

Le programme COURSE crée un lutin agrandi sur quatre caractères (codes 32, 33, 34 et 35), ce qui nécessite que vous appuyez sur n'importe quelle touche au départ.
A l'aide des touches, vous pouvez ensuite modifier la vitesse pour faire accélérer ou ralentir la voiture. Après l'arrêt, tapez CALL LOAD(−31878,0) pour faire disparaître tout lutin. A tout instant, les adresses 768 et 769 contiennent les positions verticales et horizontales du lutin (coin supérieur gauche).

Remarques complémentaires

Si vous ne possédez pas la Mini-mémoire mais le module Editeur/Assembleur, tout ce qui a été dit reste valable, sauf dans la section "Fichiers en Mini-mémoire", où le fichier devrait être créé sur disquette.

Calcul des adresses décimales

Pour convertir une adresse hexadécimale en adresse décimale utilisable pour CALL LOAD ou CALL PEEK, faire la conversion hexadécimale-décimale. Si le résultat est supérieur à 32767, soustraire 65536, sinon laisser tel quel.
Chaque adresse représente la position en mémoire d'un octet (8 bits). ■

Communiqué de Texas Instruments France

1) Texas Instruments France confirme l'arrêt de la fabrication du TI-99/4A, depuis novembre dernier.
Depuis Noël 83, suite aux ruptures de stock, Texas Instruments France s'efforce d'identifier d'éventuels stocks de matériels restant chez des revendeurs européens, et de les mettre à disposition d'autres revendeurs français en ayant fait la demande, ceci après vérifications et mises aux normes locales (manuel, transformateur d'alimentation, sortie SECAM ou Péritel).

Le consommateur pourra encore trouver quelques consoles, manettes de jeu et modules, notamment :

Mini-mémoire
Basic Etendu
Othello
Amazing
Tombstone City
Foot Ball (français)
Parsec
TI-Invaders
Blastoo
Car Wars
The Attack
Munch-Man

Quelques nouveaux logiciels, annoncés par Texas Instruments au SICOB dernier, vont quand même voir le jour :

Jawbreaker
Demon attack
Gestion privée
Mash
Hopper
Burger time
Tresure island
Star treck
Retour du pirate

Extensions périphériques

Elles sont introuvables, les possesseurs de TI-99 s'étant littéralement jetés dessus dès l'annonce de l'arrêt de la fabrication, bouleversant en cela toutes les prévisions.

2) Par ailleurs, nous pouvons confirmer que Texas Instruments France a bien porté sa garantie, sur la ligne TI-99/4A, de six mois à un an, pour les achats faits à partir du premier novembre 1983.
En outre, Texas Instruments confirme la poursuite de tous les moyens (équipe technique et pièces détachées) pour continuer le service après vente pendant plusieurs années, au-delà de la période de garantie.

***NDLR** : d'après certaines rumeurs, Texas Instruments aurait cédé plusieurs licences à des constructeurs indépendants, dans le but d'améliorer la situation des utilisateurs du TI-99/4A. Si tel est le cas, nous pouvons nous attendre à voir arriver quelques nouveaux logiciels.*
D'autre part, les modules annoncés partiront certainement très rapidement, nous ne pouvons donc que vous conseiller de "guetter" l'arrivée de ces logiciels chez votre revendeur (il serait peut-être prudent de réserver). ■

Le serveur de restaurant

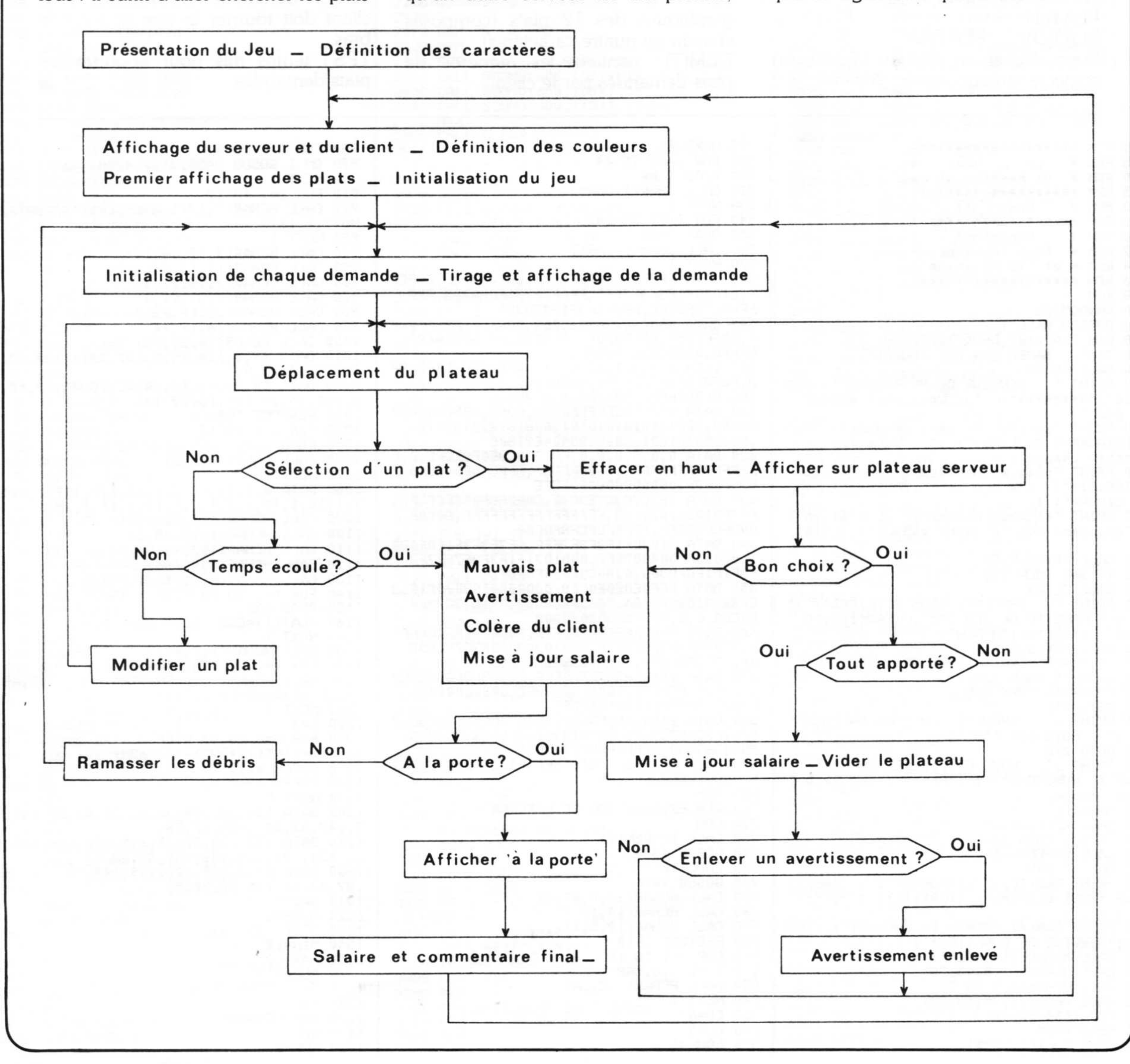
Edouard Krok

***NDLR** : ce programme occupe une grande partie de la mémoire vive, et n'est donc pas utilisable directement par les lecteurs munis d'un contrôleur de disquette (voir l'article de Denise Amrouche, dans ce numéro). Si tel est votre cas, vous devrez, avant de charger le programme en mémoire, taper CALL FILES(1) puis NEW. Il est ensuite nécessaire de supprimer toutes les lignes de REMarques. Après cela, le programme fonctionne normalement.*

Ce jeu écrit en **Basic TI** convient à tous : il suffit d'aller chercher les plats ou couverts demandés par le client. Mais bientôt, le client devient plus exigeant.

"... que m'a-t'il demandé ? De la salade ou de la purée ? Je ne me souviens plus ! Ce dont je suis sûr, c'est qu'il m'a commandé de la saucisse, mais j'ai beau guetter les plats qui arrivent de la cuisine, il n'y en a pas ! Le client va finir par se lasser d'attendre ! Peut-être se plaindra-t-il au patron du restaurant ? ... et toujours pas de saucisses !

Soudain, je les vois là-bas, au fond ! Je me précipite pour arriver avant qu'un autre serveur ne les prenne; j'attrappe ce plat tant attendu et je m'empresse de l'amener au client. Par comble de malchance, dans ma précipitation, ce n'est pas le plat de saucisses que j'ai attrapé, mais un steak qui était à côté. Cela n'a pas l'air de plaire à mon client qui se met à vociférer et entre dans une colère noire, tout en donnant de grands coups de poings sur la table qui ne tarde pas à se briser, cependant que le couvert vole en éclats. J'entends quelqu'un dans la salle siffler ironiquement : "Meunier, tu dors...", juste pour me faire remarquer que je manque de vigilance... quel métier !



Le patron, appelé, me donne un avertissement et me supprime 10% de mon salaire. Je vais essayer de me rattraper : peut-être m'enlèvera-t-il cet avertissement.
Sinon, au quatrième, je serai mis à la porte et il ne me restera plus qu'à chercher une place dans un autre restaurant."

Pour comprendre le fonctionnement du programme

Il y a 12 plats (ou couverts), composés chacun de quatre caractères graphiques, soit un total de 4★12=48 caractères, qui sont placés dans la matrice PLAT(48) selon un ordre précis :

1	2		5	6		9	10	>	45	46
3	4		7	8		11	12		47	48

Pour déterminer un plat, on se sert dans le programme de :
PLAT(1), PLAT(5), PLAT(9),......PLAT(45).
Pour afficher un plat sur l'écran, on respecte toujours l'ordre suivant :
3, 4, 1, 2 - 7, 8, 5, 6 - 11, 12, 9, 10..........47, 48, 45, 46.

La matrice DEM(7) sert à savoir quels sont les plats qui ont été demandés par le client, tandis que la matrice PRES(8) sert à savoir quels sont les plats qui sont présentés sur l'écran. Toutes deux ne peuvent contenir que des 0 ou l'un des 12 numéros suivants :
1 - 5 - 9 - 13 - 17 - 21 - 25 - 29 - 33 - 37 - 41 - 45.
Ces numéros sont déterminés au hasard par la formule :
(INT(12★RND)+1)★4-3.
Ils suffisent pour retrouver le plat désiré et éventuellement les graphismes qui le composent dans PLAT(No), PLAT(No+1), PLAT(No+2), PLAT(No+3).

Principales variables

PLAT(48) : contient les caractères graphiques des 12 plats (composés chacun de quatre caractères).
DEM(7) : contient les numéros de plats demandés par le client.
PRES(8) : contient les numéros des plats présents sur l'écran.
APP : compte le nombre de plats apportés.
AVER : compte le nombre d'avertissements.
CC et CL : correction colonne et correction ligne. Ces variables permettent d'afficher le serveur, "en service" et "à la porte".
MAX : nombre maximum de plats pouvant être demandés.
MIN : nombre minimum de plats pouvant être demandés.
MSAL : salaire maximum.
OT : compte le nombre de plats apportés depuis le dernier avertissement. Sert à enlever un avertissement.
RECH : contient le numéro de la colonne où se trouve le petit plateau qui va chercher les plats.
SAL : salaire du serveur.
TETE : détermine (au hasard) si le client doit tourner la tête et lever le bras.
TPS : temps mis pour apporter les plats demandés. ■

```
100 REM ******************
105 REM *   Le  serveur   *
110 REM *  de restaurant  *
115 REM ******************
120 REM *    Basic  TI    *
125 REM ******************
130 REM *    Copyright    *
132 REM *  Edouard  Krok  *
134 REM * et "99 Magazine"*
136 REM ******************
138 REM
140 RANDOMIZE
150 OPTION BASE 1
160 DIM PLAT(48),DEM(7),PRES(8)
170 T$="   TAPEZ SUR UNE TOUCHE"
180 CALL CLEAR
190 PRINT "   SERVEUR DE RESTAURANT": :TA
B(10);"********": :"   Vous etes embauc
he pour": :
200 PRINT "assurer le service dans": :"un
 RESTAURANT de LUXE.": :"   Vous irez c
hercher chaque": :
210 PRINT "plat en deplacant le plateau":
 :TAB(8);">  : a droite": :TAB(8);"<  :
 a gauche": :
220 PRINT "et vous choisirez le plat en":
 :"appuyant sur la barre espace": : :T$
;
230 CALL KEY(3,K,S)
240 IF S=0 THEN 230
250 CALL CLEAR
260 PRINT "   PREFEREZ VOUS UTILISER": :"
  UNE MANETTE DE JEU ?": : :TAB(10);"O
_OUI": :TAB(10);"N_NON": :
270 CALL KEY(5,K,S)
280 IF (K=78)+(K=110)THEN 380
290 IF K=79 THEN 320
300 IF K=111 THEN 340
310 GOTO 270
320 PRINT :"   APPUYEZ SUR ""ALPHA LOCK""
": :"   PUIS SUR ""O"""
330 GOTO 270
340 PRINT :"   VOUS UTILISEREZ LA": :"
MANETTE NUMERO 2": : :"Appuyez sur le b
outon rouge"
350 MAN=1
360 CALL KEY(2,K,S)
370 IF S=0 THEN 360
380 CALL SCREEN(15)
390 CALL CLEAR
400 PRINT TAB(10);"RESTAURANT": : :TAB(7)
;"LA TOQUE BLANCHE": :TAB(11);"!!!!!!!!
": : : : :
410 PRINT TAB(7);"MENUS DE CHOIX": : :TAB
(7);"SERVICE DE QUALITE": : : : :
420 CALL HCHAR(3,5,45,25)
430 CALL HCHAR(22,5,45,25)
440 CALL VCHAR(3,5,45,20)
450 CALL VCHAR(3,29,45,20)
460 FOR I=34 TO 46
470 READ CAR$
480 CALL CHAR(I,CAR$)
490 NEXT I
500 FOR I=58 TO 64
510 READ CAR$
520 CALL CHAR(I,CAR$)
530 NEXT I
540 FOR I=91 TO 149
550 READ CAR$
560 CALL CHAR(I,CAR$)
570 NEXT I
580 DATA FFFFFFFFFFFFFFFF,80C0F0F8FCFCFEF
F,01030F1F3F3F7FFF,FFFF7F7F1F0F0703,FFF
FFEFEF8F0E0C,FF7F3F3F1F0F0301
590 DATA 000C3063820 60C08,00FFFFFFFFFFFFF
F,00FFFFFF,0F0F0F0F0F0FFF3F,F4FAF2F4F8F
0FFFC,0000002A2A2A3E1C
600 DATA 08081C1C1C1C1C1C,FFFFFFFFF8F8F8F
8,F8F8F8F8FFFFFFFF,80008000801F8F87,000
0000000FFFFFF,0001000100F9F0E1
610 DATA EEEEEEEEEEEEEEEE,00FE,0080808080
80808,0001010101010101,0081818181818181
,008080601E01,C0600804 04E21805
620 DATA 0,0,0,0,0,0,F8F8E0C8E0F8FCFE,0,F
FFFFFFFFFFFFFFF,1F3FFFF8F,FFFC,00000000
030F3F7F,00000000C0F0FCFE
630 DATA 7E3C7E3C7E3C7E,C0E0F0F0F8FCFCFC,
FF7F010301030103,FFFFFFFFFFFFFFFF,80C0F
0F8FCFCFEFF,FFFEFCFCF8F0C08
640 DATA 01030F1F3F3F7FFF,FF7F3F3F1F0F030
1,000000000007EFF,0001071F1F3F3F7F,7F7
F3F1F1F0F0303,0080C0E0E0F0F8FC
650 DATA FCF0E0E0E0C08,0000000103070F1E,3
CF860A1030F060A,5060F0C0C5860F1C,3C78F0
E0C08,0,0,FFFF808080808080
660 DATA FFFFFFFFFFFFFFFF,80C0F0F8FCFCFEF
F,01030F1F3F3F7FFF,000103030F0F3F7F,80C
0E0F0FCFCFEFF,0818183838383818
670 DATA 1818183C7EFFFFFF,7E3C3C3C3C3C3C7
E,00FFFFFFFFFF,0000F0F8F8FC,0000000001
E3E7F,7FFFFFFF3F1F1F0F
680 DATA E0E0F0F0F0E0C,0000040C1C1C3C3C,7
C7C7C7C7C7C7C7C,0810102026 4F5F86,383838
3838383838,FFFFFFFFF7E7E3C
690 DATA 18181818181818FF,000C0E0687E77F3
F,80C0CCCCECEFFEFE
700 REM
710 REM PREPARATION DE L'ECRAN
720 REM
730 CALL SCREEN(2)
740 GOSUB 4530
750 COUL=16
760 GOSUB 4460
770 CALL HCHAR(14,8,42,8)
780 CALL HCHAR(14,10,41,4)
790 CALL VCHAR(19,5,112,5)
800 RESTORE 1010
810 FOR I=1 TO 11
820 READ L,C,CAR
830 CALL HCHAR(L,C,CAR)
840 NEXT I
850 CC=0
860 CL=0
870 COUL=1
880 FOR I=1 TO 4
890 ON I GOSUB 3450,4100,4240,4460
900 NEXT I
910 FOR I=1 TO 12
920 CALL HCHAR(8,17+I,ASC(SEG$("APPORTEZ
MOI",I,1)))
930 NEXT I
940 CALL HCHAR(7,17,64,14)
950 CALL HCHAR(11,17,64,14)
960 CALL VCHAR(7,16,92,4)
970 CALL VCHAR(7,31,91,4)
980 CALL VCHAR(11,19,93,3)
990 CALL HCHAR(14,19,94)
1000 CALL HCHAR(14,20,95)
1010 DATA 19,6,114,24,5,137,24,6,138,13,2
3,117,14,23,98
1020 DATA 14,22,96,14,24,97,15,23,99,15,2
2,100,15,24,101,16,23,136
1030 RESTORE 1080
1040 FOR I=1 TO 15
1050 READ L,C
1060 CALL COLOR(I,L,C)
1070 NEXT I
1080 DATA 6,1,14,1,12,6,12,6,1,1,1,1,1,1,
1,1,10,1,10,1,7,1,7,1,12,1,11,1,13,1
1090 CALL HCHAR(1,1,32,192)
1100 CALL HCHAR(1,25,48,3)
1110 CALL HCHAR(1,7,48,3)
1120 CALL HCHAR(4,3,136)
1130 CALL HCHAR(4,31,136)
1140 FOR I=1 TO 48
1150 READ CAR
1160 PLAT(I)=CAR
1170 NEXT I
1180 CALL HCHAR(1,7,48,3)
1190 FOR I=1 TO LEN(STR$(MSAL))
1200 CALL HCHAR(1,9-LEN(STR$(MSAL))+I,ASC
(SEG$(STR$(MSAL),I,1)))
1210 NEXT I
1220 L=4
1230 FOR C=6 TO 27 STEP 3
1240 W=(INT(12*RND)+1)*4-3
1250 GOSUB 4590
1260 PRES(C/3-1)=W
1270 NEXT C
1280 DATA 134,32,145,32,45,32,46,32,146,3
2,147,32,107,108,37,38
1290 DATA 132,133,37,38,118,120,119,121,1
22,124,123,125,148,149,37,38
1300 DATA 36,35,39,128,139,144,140,141,32
,32,43,44,135,32,59,32
1310 SAL=0
1320 AVER=0
1330 OT=0
1340 MIN=.7
1350 MAX=.2
1360 RECH=15
1370 CALL HCHAR(5,RECH,42,2)
1380 REM
1390 REM COMMANDE DU CLIENT
1400 REM
1410 APP=0
```

```
1420 TPS=0
1430 IF MIN>6 THEN 1450
1440 MIN=MIN+.3
1450 IF MAX>6.2 THEN 1470
1460 MAX=MAX+.8
1470 QDEM=INT(((MAX-MIN+1)*RND)+MIN)
1480 FOR I=1 TO QDEM
1490 DEM(I)=(INT(12*RND)+1)*4-3
1500 NEXT I
1510 TETE=INT(6*RND)+1
1520 IF TETE<4 THEN 1540
1530 GOSUB 3470
1540 COUL=16
1550 GOSUB 4460
1560 CALL SOUND(500,800,10,-7,10)
1570 L=10
1580 FOR I=1 TO QDEM
1590 C=15+I*2
1600 W=DEM(I)
1610 GOSUB 4590
1620 NEXT I
1630 FOR DELAI=1 TO QDEM*20+50
1640 CALL KEY(2,K,S)
1650 IF S<>0 THEN 1670
1660 NEXT DELAI
1670 IF TETE<4 THEN 1690
1680 GOSUB 3660
1690 CALL HCHAR(9,17,32,14)
1700 CALL HCHAR(10,17,32,14)
1710 COUL=1
1720 GOSUB 4460
1730 CALL SOUND(200,1000,5)
1740 REM
1750 REM  DEPLACEMENT
1760 REM
1770 IF MAN THEN 1890
1780 CALL KEY(3,K,S)
1790 IF K=44 THEN 1830
1800 IF K=46 THEN 1860
1810 IF K=32 THEN 2120
1820 GOTO 2000
1830 IF RECH=6 THEN 2000
1840 RECH=RECH-3
1850 GOTO 1980
1860 IF RECH=27 THEN 2000
1870 RECH=RECH+3
1880 GOTO 1980
1890 CALL KEY(2,K,S)
1900 IF K=18 THEN 2120
1910 CALL JOYST(2,X,Y)
1920 IF X=0 THEN 2000
1930 RECH=RECH+3*X/4
1940 IF RECH>4 THEN 1960
1950 RECH=6
1960 IF RECH<28 THEN 1980
1970 RECH=27
1980 CALL HCHAR(5,3,32,29)
1990 CALL HCHAR(5,RECH,42,2)
2000 L=4
2010 C=(INT(8*RND)+1)*3+3
2020 IF C=RECH THEN 1780
2030 W=(INT(12*RND)+1)*4-3
2040 TPS=TPS+1
2050 IF TPS=QDEM*8+30 THEN 2320
2060 GOSUB 4590
2070 PRES(C/3-1)=W
2080 GOTO 1770
2090 REM
2100 REM CHOIX D'UN PLAT
2110 REM
2120 CALL SOUND(200,-5,4)
2130 IF PRES(RECH/3-1)=0 THEN 1780
2140 CALL HCHAR(3,RECH,32,2)
2150 CALL HCHAR(4,RECH,32,2)
2160 APP=APP+1
2170 IF APP>4 THEN 2210
2180 L=13
2190 C=6+APP*2
2200 GOTO 2240
2210 L=10
2220 C=APP*2-1
2230 CALL HCHAR(11,9,42,6)
2240 W=PRES(RECH/3-1)
2250 GOSUB 4590
2260 FOR I=1 TO QDEM
2270 IF DEM(I)=PRES(RECH/3-1)THEN 2570
2280 NEXT I
2290 REM
2300 REM  MAUVAIS PLAT
2310 REM
2320 CALL SOUND(300,294,3)
2330 CALL SOUND(600,392,3)
2340 CALL SOUND(300,494,3)
2350 CALL SOUND(900,392,3)
2360 PRES(RECH/3-1)=0
2370 GOSUB 3820
2380 FOR I=310 TO 110 STEP -50
2390 CALL SOUND(300,I*2,1,500,6)
2400 CALL SOUND(500,I,1,300,6)
2410 NEXT I
2420 SAL=INT(SAL*.9)
2430 OT=0
2440 GOSUB 4670
2450 AVER=AVER+1
2460 IF AVER>3 THEN 2830
2470 CALL HCHAR(1,13+AVER*2,63)
2480 FOR L=9 TO 13
2490 CALL HCHAR(L,8,32,8)
2500 NEXT L
2510 GOSUB 4070
2520 GOSUB 3450
2530 GOTO 1410
2540 REM
2550 REM  BON PLAT
2560 REM
2570 CALL SOUND(100,1000,6)
2580 PRES(RECH/3-1)=0
2590 DEM(I)=0
2600 OT=OT+1
2610 IF APP<QDEM THEN 1780
2620 REM
2630 REM  TOUT APPORTE
2640 REM
2650 CALL SOUND(300,880,1,440,4,1760,2)
2660 CALL SOUND(150,784,1,392,4,784,2)
2670 CALL SOUND(450,784,1,392,4,784,2)
2680 SAL=SAL+APP*10
2690 GOSUB 4670
2700 FOR L=9 TO 13
2710 CALL HCHAR(L,8,32,8)
2720 NEXT L
2730 IF AVER=0 THEN 1410
2740 IF OT<10 THEN 1410
2750 IF TPS>50 THEN 1410
2760 CALL HCHAR(1,13+AVER*2,32)
2770 AVER=AVER-1
2780 OT=0
2790 GOTO 1410
2800 REM
2810 REM  A LA PORTE
2820 REM
2830 FOR I=50 TO 0 STEP -50
2840 CALL SOUND(600,150+I,1)
2850 CALL SOUND(600,180+I,1)
2860 CALL SOUND(600,110+I,1)
2870 NEXT I
2880 CALL CLEAR
2890 PRINT "      VOUS ETES RENVOYE": : :
: : : : : : : : : : : : : : : : : : : :
2900 COUL=16
2910 GOSUB 4460
2920 CALL VCHAR(4,4,34,17)
2930 CALL VCHAR(4,5,34,17)
2940 CALL VCHAR(4,13,34,17)
2950 CALL VCHAR(4,14,34,17)
2960 CALL HCHAR(4,4,34,11)
2970 CALL HCHAR(4,4,34,11)
2980 GOSUB 4370
2990 CL=-2
3000 CC=16
3010 GOSUB 4240
3020 CL=3
3030 GOSUB 4370
3040 CALL HCHAR(17,21,112,2)
3050 CALL HCHAR(18,21,116)
3060 CALL HCHAR(18,20,40)
3070 CALL HCHAR(19,19,40,2)
3080 PRINT T$
3090 CALL KEY(2,K,S)
3100 IF S=0 THEN 3090
3110 REM
3120 REM COMMENTAIRE FINAL
3130 REM
3140 FOR I=1 TO 16
3150 CALL COLOR(I,2,1)
3160 NEXT I
3170 CALL CLEAR
3180 CALL SCREEN(15)
3190 IF MSAL>SAL THEN 3210
3200 MSAL=SAL
3210 PRINT "VOTRE SALAIRE ";SAL;"FRANCS":
 : : : : :
3220 IF SAL>500 THEN 3340
3230 IF SAL>300 THEN 3320
3240 IF SAL>200 THEN 3300
3250 IF SAL>80 THEN 3280
3260 PRINT "   JE VOUS CONSEILLE": : :"
 DE CHANGER DE METIER"
3270 GOTO 3350
3280 PRINT "   VOUS AVEZ BESOIN": : :"
D APPRENDRE LE METIER"
3290 GOTO 3350
3300 PRINT "   EXERCEZ VOTRE MEMOIRE": :
:"   ET REVENEZ ME VOIR"
3310 GOTO 3350
3320 PRINT "   SI VOUS LE DESIREZ": : :"
  JE PEUX VOUS REEMBAUCHER"
3330 GOTO 3350
3340 PRINT " VOUS ETES REEMBAUCHE": : :"
JE VOUS SUPPLIE D ACCEPTER"
3350 PRINT : : : : : : : : :T$
3360 CALL KEY(2,K,S)
3370 IF S=0 THEN 3360
3380 GOTO 730
3390 REM
3400 REM AFFICHAGE DU SERVEUR
3410 REM       ET DU CLIENT
3420 REM
3430 CALL CHAR(103,"FFFFF8F8C09838F")
3440 RETURN
3450 CALL CHAR(103,"FF7FBC98C0C0F8F8")
3460 RETURN
3470 CALL CHAR(96,"01070104010F3F3F")
3480 CALL CHAR(97,"0080E060B0B0B878")
3490 CALL CHAR(98,"FFFFFFFCFBFEFFFF")
3500 CALL CHAR(100,"FF1F000001031F03")
3510 CALL CHAR(99,"DE3F7FFFFFFFFFFF")
3520 CALL CHAR(101,"F8F8F8F0F0F0C08")
3530 IF TETE<6 THEN 3650
3540 CALL VCHAR(17,20,32,3)
3550 CALL HCHAR(19,21,32)
3560 CALL HCHAR(17,21,34)
3570 CALL HCHAR(18,21,39)
3580 CALL HCHAR(16,20,35)
3590 CALL HCHAR(17,20,39)
3600 CALL HCHAR(15,19,35)
3610 CALL HCHAR(16,19,39)
3620 CALL HCHAR(15,18,39)
3630 CALL HCHAR(14,18,110)
3640 CALL HCHAR(14,17,111)
3650 RETURN
3660 IF TETE<6 THEN 3750
3670 FOR W=13 TO 16
3680 CALL VCHAR(W,W+4,32,2)
3690 NEXT W
3700 CALL HCHAR(17,21,36)
3710 CALL VCHAR(18,21,34,2)
3720 CALL HCHAR(19,20,109)
3730 CALL HCHAR(18,20,145)
3740 CALL HCHAR(17,20,134)
3750 CALL CHAR(96,"0103072E3C2E3F1F")
3760 CALL CHAR(97,"80C0E0743C74FCF8")
3770 CALL CHAR(98,"FFFFFF3C993CFFFF")
3780 CALL CHAR(99,"FFFFFF3C81C3FFFF")
3790 CALL CHAR(100,"1F1D3E3F3F1F0701")
3800 CALL CHAR(101,"F8B87CFCFCF8E08")
3810 RETURN
3820 CALL CHAR(99,"FFC300000018FFFF")
3830 CALL CHAR(100,"1F1F3F3C381C0701")
3840 CALL CHAR(101,"F8F8FC3C1C38E08")
3850 GOSUB 3430
3860 CALL VCHAR(17,20,32,3)
3870 CALL HCHAR(19,21,39)
3880 CALL HCHAR(19,22,35)
3890 CALL HCHAR(20,22,39)
3900 CALL HCHAR(20,23,35)
3910 CALL HCHAR(21,23,39)
3920 CALL HCHAR(20,17,32,3)
3930 FOR W=22 TO 18 STEP -1
3940 CALL HCHAR(W,W+1,113)
3950 CALL HCHAR(W+1,W+1,116)
3960 NEXT W
3970 CALL HCHAR(21,24,109)
3980 CALL HCHAR(19,23,40,2)
3990 CALL HCHAR(20,24,40,2)
4000 CALL HCHAR(17,28,40)
4010 CALL HCHAR(20,28,32,2)
4020 FOR W=25 TO 28
4030 CALL HCHAR(46-W,W,115)
4040 CALL HCHAR(47-W,W,114)
4050 NEXT W
4060 RETURN
4070 FOR W=23 TO 17 STEP -1
4080 CALL HCHAR(W,19,32,10)
4090 NEXT W
4100 CALL VCHAR(21,19,145,3)
4110 CALL VCHAR(21,27,145,3)
4120 CALL HCHAR(17,22,34,3)
4130 CALL HCHAR(18,21,34,5)
4140 CALL HCHAR(20,17,112,13)
4150 RESTORE 4220
4160 FOR W=1 TO 11
4170 READ L,C,CAR
4180 CALL HCHAR(L,C,CAR)
4190 NEXT W
4200 GOSUB 3700
4210 RETURN
4220 DATA 19,22,60,19,23,61,19,24,62,19,2
5,34,19,26,109,17,25,35
4230 DATA 18,26,46,17,26,45,19,28,59,18,2
8,58,17,28,135
4240 RESTORE 4340
4250 FOR W=4 TO 6
4260 CALL VCHAR(14+CL,W+CC,63,5)
4270 NEXT W
4280 CALL HCHAR(15+CL,7+CC,137,3)
4290 FOR W=1 TO 16
4300 READ L,C,CAR
4310 CALL HCHAR(L+CL,C+CC,CAR)
4320 NEXT W
4330 RETURN
4340 DATA 19,4,116,13,5,129,13,4,131,13,6
,130,12,5,136,11,5,104
4350 DATA 10,5,104,11,4,119,10,4,112,9,4,
118,9,5,112,9,6,120
4360 DATA 10,6,102,11,6,103,15,10,105,15,
11,106
4370 FOR W=14 TO 17
4380 CALL HCHAR(W+CL,W-8+CC,113)
4390 CALL HCHAR(W+1+CL,W-8+CC,112)
4400 CALL HCHAR(W+2+CL,W-8+CC,116)
4410 NEXT W
```

```
4420 CALL HCHAR(19+CL,9+CC,32)
4430 CALL VCHAR(17+CL,10+CC,143,2)
4440 CALL HCHAR(16+CL,10+CC,142)
4450 RETURN
4460 FOR W=5 TO 8
4470 CALL COLOR(W,COUL,1)
4480 NEXT W
4490 RETURN
4500 REM
4510 REM MESSAGE D'ATTENTE
4520 REM
4530 CALL CLEAR
4540 PRINT TAB(7);"TENEZ VOUS PRET": : :"
  LES PREMIERS CLIENTS": :"   NE DEVRA
IENT PAS TARDER": : : : : : : : : : : :
 : : : : :
4550 RETURN
4560 REM
4570 REM AFFICHAGE DES PLATS
4580 REM
4590 CALL HCHAR(L,C,PLAT(W+2))
4600 CALL HCHAR(L,C+1,PLAT(W+3))
4610 CALL HCHAR(L-1,C,PLAT(W))
4620 CALL HCHAR(L-1,C+1,PLAT(W+1))
4630 RETURN
4640 REM
4650 REM AFFICHAGE DU SALAIRE
4660 REM
4670 CALL HCHAR(1,25,48,3)
4680 FOR W=1 TO LEN(STR$(SAL))
4690 FOR I=48 TO ASC(SEG$(STR$(SAL),W,1))
4700 CALL HCHAR(1,27-LEN(STR$(SAL))+W,I)
4710 CALL SOUND(100,W*I*10,5)
4720 NEXT I
4730 NEXT W
4740 RETURN
```

Le module "Echecs" à l'essai

Alain Paloma

Dans la catégorie des jeux de réflexion, Texas Instruments commercialise sous forme de module un magnifique programme de jeu d'échecs.

Ma première surprise fut de constater qu'après avoir introduit le module, l'habituelle mire indiquant :

READY - PRESS ANY KEY TO BEGIN

était devenue :

POUR COMMENCER APPUYEZ SUR N'IMPORTE QUELLE TOUCHE

L'écran suivant est aussi en français :

ORDINATEUR FAMILIAL
CHOISISSEZ :
1-UTILISATION EN "BASIC"
2-CALCUL D'UNE EQUATION
3-JEU D'ECHECS

(Que vient donc faire ici le calcul d'une équation, serait-ce un nouveau "bug" ?).

Ceci n'a certes que peu de rapport avec les échecs, mais méritait cependant d'être souligné.

Venons-en au programme d'échecs proprement dit. Avant d'aller plus loin dans l'exploration de ce logiciel, mettons les choses au point. Ce module permet aux débutants de s'initier rapidement et efficacement aux échecs et aux joueurs confirmés d'avoir, toujours disponible, un partenaire de choix, mais qui ne peut en aucune façon mettre sérieusement en danger un adversaire de haut niveau. Au niveau 2 (il y en a 3 au total), un joueur moyen (en l'occurence votre dévoué serviteur) peut arriver à le battre (je n'y suis jamais parvenu au niveau supérieur). Ceci étant dit, jouons...



Une fois de plus, le graphisme est très soigné (nous avions fait la même remarque pour le module "Othello"), les pièces sont finement dessinées et les couleurs assez judicieusement choisies. Une petite musique d'accompagnement se fait entendre à chaque déplacement d'une pièce (les pièces glissent sur l'échiquier, ce qui laisse supposer qu'il s'agit de "lutins").

A gauche de l'échiquier apparaissent deux chronomètres, qui comptabilisent les temps de réflexion des deux adversaires. Il est possible de jouer contre un "humain", le TI-99 étant alors réduit à la seule fonction d'échiquier, ou contre lui; il fait alors preuve, selon le choix effectué en début de partie, d'un comportement agressif, défensif ou normal. Le temps de réflexion de l'ordinateur est directement lié au niveau de jeu.

La principale attraction de ce module, outre le fait qu'il fournit un bon adversaire, est qu'il comporte une option permettant au débutant de s'initier à la stratégie des échecs.
Les options de jeu sont nombreuses : il est, par exemple, possible de permuter l'échiquier (le TI-99 prend votre jeu et vous le sien) à tout moment. La partie étant intégralement mémorisée, il est également possible de revenir en arrière d'autant de phases de jeu qu'on le désire, et de reprendre la partie (ceci est très utile quand la situation est désespérée, pour reculer l'échéance fatale).

On peut évidemment soumettre un problème à l'ordinateur, lui demander conseil en cours de partie, etc... On peut même jouer plusieurs parties simultanées : l'ordinateur s'occupe de tout.
Enfin, lorsque vous saurez que l'on peut sauvegarder (sur cassette exclusivement) une partie en cours, afin de pouvoir la reprendre ultérieurement, vous pourrez vous précipiter chez votre revendeur. ■

Remise à zéro des variables en Pascal

En Pascal, comme en Assembleur (et contrairement au Basic), lors de l'exécution d'un programme, aucune mémoire n'est remise à zéro. Cette propriété, qui est souvent très intéressante (notamment pour situer les erreurs dues aux débordements), peut parfois provoquer des erreurs pour le moins inattendues si l'on conserve ses vieux automatismes "Basiciens". N'oubliez donc jamais d'initialiser toutes vos variables, et particulièrement les tableaux, qui sont généralement les premières victimes. ■

Combat de chars

Jean-Claude Hoet

C'est à la vue de la cassette de jeu "Blastoo" que l'envie de créer un tel programme m'est venue.
Vous verrez que le jeu est très différent de son grand frère, mais que son principal atout réside dans le réalisme de l'action et dans l'intense tension que ce jeu génère chez les deux joueurs, l'ordinateur se contentant de fournir le décor de la bataille.

Comment fonctionne "Combat de chars" ?

Tout d'abord, le charger; et pour cela la configuration de base suffit puisqu'il est écrit en Basic TI.
Le contrôle des chars s'effectue à partir du clavier.

Après un court "copyright" destiné à occuper l'écran pendant la définition des caractères, une ville se dessine puis un "bip" sonore annonce le début des hostilités.
Au départ, les deux chars se trouvent respectivement en bas à gauche et à droite de l'écran.

Le char de gauche se commande avec les touches A, S, X et D, et le char de droite avec les touches H, J, M et K.
La touche A (H) permet de faire avancer le char considéré.
La touche S (J) permet de faire tourner le char dans le sens des aiguilles d'une montre.
La touche X (M) permet de faire tourner le char dans le sens inverse des aiguilles d'une montre.
Enfin, la touche D (K) permet le tir du canon.

Une certaine dextérité sera nécessaire avant de pouvoir guider son char sans encombre. Pour ma part, je conseille vivement la technique suivante :
Le majeur de la main gauche sur A (H), l'annulaire de la main gauche alternativement sur S (J) ou X (M) suivant la direction à prendre. Enfin, l'annulaire de la main droite sur la touche de tir D (K).

Ceux qui possèdent des manettes de jeu peuvent facilement modifier le programme pour une utilisation plus aisée, mais attention, le piment de ce jeu réside, entre autre, dans le fait qu'un char ne fait pas demi-tour instantanément (il lui faut quatre temps), ce qui permet à un adversaire qui parvient à ce placer derrière un char de faire feu ! (ce qui ne prend qu'un temps). Il faut qu'il en soit de même après transformation.

Vous remarquerez très vite, à vos dépends, qu'il faut deux obus pour perforer un mur intérieur et un seul si vous êtes directement contre.
Le mur extérieur à la ville est "indestructible" (les obus explosent au pied). Si vous tirez dessus à courte distance, votre char explose !

La simulation des explosions comprend une flamme à la sortie du canon et un caractère représentant une explosion. Ce caractère se place sur le premier obstacle rencontré et une simulation visuelle est obtenue en faisant varier alternativement et très rapidement la couleur du caractère et de son fond (rouge sur blanc). Un char touché provoque un véritable cataclysme.

La pression d'une touche permet ensuite d'afficher le tableau de chasse. Pour "avoir" régulièrement votre adversaire, une stratégie sans cesse renouvelée vous sera nécessaire, ce qui rend ce jeu très prenant, d'après les multiple amis qui ont eu l'occasion de s'y essayer.

Comment fonctionne ce programme ?

La limitation pour un tel jeu, écrit en **Basic TI**, est la vitesse d'exécution. Bien qu'il s'agisse ici d'un combat de chars et qu'un char soit plutôt lent, il m'a fallu accélérer au maximum le processus de déplacement des chars.

Ayant remarqué auparavant que les GOSUB et les boucles FOR-NEXT ralentissent considérablement le programme, j'ai opté pour une programmation "en ligne" où j'utilise au maximum les instructions ON-GOTO et GOTO, pour diriger le programme vers les parties qui traitent le mouvement demandé.

Il y a huit directions possibles par char, ce qui nous donne seize caractères (deux chars de couleurs différentes), huit caractères "flamme" et cinq caractères spéciaux (explosion, mur, ...).

136 à 143 : char blanc
144 à 151 : char noir
152 à 159 : flamme au canon
45 : explosion
46 : croix de la tombe
128 : mur
129 : mur extérieur
130 : débris

Le plan de la ville est fixé dans les lignes de DATA en fournissant les coordonnées nécessaires pour les instructions HCHAR et VCHAR (ligne, colonne, répétition). Le signe moins (-) dans les DATA indique au programme qu'il s'agit de données pour une instruction VCHAR, alors que l'absence du signe oriente les données vers une instruction HCHAR.

En gros, et pour le char blanc :

810 : entrée d'une commande. Si rien, GOTO 2080 (char noir)
820 : mouvement du char blanc
850 : le char blanc contre un mur (ne peut pas le traverser)
870 : efface le char de son ancienne position
900 : positionne le char à sa nouvelle position
1500 : tir du char blanc
1510 : qu'y a-t-il devant le char blanc ? (après le tir)
1590 : objectif atteint. Dessin de l'explosion puis de débris (ou de tombe)
1620 : explosion d'un obus sur un mur neuf
1640 : simulation de l'explosion
1700 : dessin de débris
1710 : efface la flamme du canon
1730 : explosion contre le mur extérieur
1840 : explosion d'un mur déjà touché et création d'un passage
1950 : explosion du char touché
1990 : simulation de l'explosion du char
2080 : même suite logique pour le char noir

En fin de programme se trouve la sous-routine qui permet d'afficher le tableau de chasse.
Ce programme occupe 7928 octets (sans les REMs) et 8413 après les réservations des variables. ■

```
10 REM *******************
20 REM * Combat de Chars  *
30 REM *******************
40 REM *     Basic TI     *
50 REM *******************
60 REM *    Copyright     *
70 REM * Jean-Claude Hoet *
80 REM * et "99 Magazine" *
90 REM *******************
100 CALL CLEAR
110 CALL SCREEN(6)
120 GOSUB 3570
125 REM ***************
126 REM DEF. CARACTERES
127 REM ***************
130 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
140 CALL CHAR(45,"105A3E67E67C3A48")
150 CALL CHAR(130,"BC3D01CDCCC303F0")
```

```
160 CALL CHAR(129,"FFFFC3C3C3C3FFFF")
170 CALL CHAR(46,"10107C10101038FE")
180 CALL COLOR(13,11,1)
190 DATA 11,2,5,13,2,5,14,2,5,17,3,4,4,9,
2,12,9,4,14,8,4,17,8,4,3,12,7,18,15,4,2
0,13,6
200 DATA 7,21,6,14,19,6,16,22,2,17,22,2,4
,25,6,5,25,6,17,26,6,-1,4,9,-20,4,5,-17
,6,4
210 DATA -1,6,7,-1,7,7,-14,8,3,-4,9,7,-15
,11,2,-19,10,6,-4,12,4,-9,12,3,-14,13,6
220 DATA -5,15,8,-15,15,3,-4,18,8,-8,21,4
,-1,22,5,-1,23,5,-9,24,5,-20,22,5,-20,2
3,5
230 DATA -18,26,3,-20,30,5,9,26,5,13,26,5
,-10,26,3,-10,30,3,11,31,1,0,0,0
240 CALL CHAR(136,"101010387C387C38")
250 CALL CHAR(152,"0000AA2854102810")
260 CALL CHAR(153,"2018361D36182000")
270 CALL CHAR(154,"1028546CAA540000")
280 CALL CHAR(155,"04083C78A4782C08")
290 CALL CHAR(156,"008040E070F8F4AA")
300 CALL CHAR(157,"AA7CF870E0408000")
310 CALL CHAR(158,"553E5F0E17020500")
320 CALL CHAR(159,"000502170E5F3E55")
330 CALL CHAR(144,"101010387C387C38")
340 CALL CHAR(137,"00000A1FFF1F0A00")
350 CALL CHAR(145,"00000A1FFF1F0A00")
360 CALL CHAR(138,"387C387C38101010")
370 CALL CHAR(146,"387C387C38101010")
380 CALL CHAR(139,"000050F8FFF85000")
390 CALL CHAR(140,"8054387D3E5F0E14")
400 CALL CHAR(141,"012A1CBE7CFA7028")
410 CALL CHAR(142,"140E5F3E7D385480")
420 CALL CHAR(143,"2870FA7CBE1C2A01")
430 CALL CHAR(147,"000050F8FFF85000")
440 CALL CHAR(148,"8054387D3E5F0E14")
450 CALL CHAR(149,"012A1CBE7CFA7028")
460 CALL CHAR(150,"140E5F3E7D385480")
470 CALL CHAR(151,"2870FA7CBE1C2A01")
480 CALL COLOR(14,16,1)
490 CALL COLOR(15,2,1)
500 CALL COLOR(16,9,1)
510 CALL CLEAR
515 REM ************
516 REM DESSIN VILLE
517 REM ************
520 RESTORE
530 READ A,B,C
540 IF A=0 THEN 610
550 CALL SOUND(-50,ABS(A*110),2)
560 IF A<0 THEN 590
570 CALL HCHAR(A,B,128,C)
580 GOTO 530
590 CALL VCHAR(-A,B,128,C)
600 GOTO 530
610 CALL HCHAR(1,2,129,31)
620 CALL HCHAR(24,2,129,31)
630 CALL VCHAR(1,2,129,24)
640 CALL VCHAR(1,32,129,24)
650 CALL SOUND(500,500,0)
655 REM **************
656 REM INITIALISATION
657 REM **************
660 A=23
670 D=23
680 DD=-1
690 AA=-1
700 B=3
710 E=31
720 EE=0
730 BB=0
740 C=136
750 F=144
760 G(1)=152
770 G(2)=152
780 CALL HCHAR(D,E,144)
790 CALL HCHAR(A,B,136)
794 REM *************
795 REM ACTION CHAR 1
796 REM *************
800 CALL KEY(1,ASA,POL)
810 IF (POL=0)+(ASA>3)THEN 2080
820 IF ASA<>1 THEN 920
830 CALL GCHAR(A+AA,B+BB,TE)
840 IF TE<128 THEN 870
850 CALL SOUND(200,1000,4)
860 GOTO 2080
870 CALL HCHAR(A,B,127)
880 A=A+AA
890 B=B+BB
900 CALL HCHAR(A,B,C)
910 GOTO 2080
920 IF ASA=3 THEN 1500
930 IF ASA=2 THEN 950
940 ON AA+2 GOTO 960,1160,1300
950 ON AA+2 GOTO 970,1170,1310
960 ON BB+2 GOTO 1320,1180,980
970 ON BB+2 GOTO 980,1040,1100
980 AA=-1
990 BB=0
1000 G(1)=152
1010 C=136
1020 CALL HCHAR(A,B,C)
1030 GOTO 2080
1040 AA=-1
1050 BB=1
1060 G(1)=156
1070 C=141
1080 CALL HCHAR(A,B,C)
1090 GOTO 2080
1100 AA=0
1110 BB=1
1120 G(1)=155
1130 C=139
1140 CALL HCHAR(A,B,C)
1150 GOTO 2080
1160 ON BB+2 GOTO 1380,2080,1040
1170 ON BB+2 GOTO 1180,2080,1240
1180 AA=-1
1190 BB=-1
1200 G(1)=159
1210 C=140
1220 CALL HCHAR(A,B,C)
1230 GOTO 2080
1240 AA=1
1250 BB=1
1260 G(1)=157
1270 C=143
1280 CALL HCHAR(A,B,C)
1290 GOTO 2080
1300 ON BB+2 GOTO 1440,1240,1100
1310 ON BB+2 GOTO 1320,1380,1440
1320 AA=0
1330 BB=-1
1340 G(1)=153
1350 C=137
1360 CALL HCHAR(A,B,C)
1370 GOTO 2080
1380 AA=1
1390 BB=-1
1400 G(1)=158
1410 C=142
1420 CALL HCHAR(A,B,C)
1430 GOTO 2080
1440 AA=1
1450 BB=0
1460 G(1)=154
1470 C=138
1480 CALL HCHAR(A,B,C)
1490 GOTO 2080
1500 CALL SOUND(300,-5,0)
1510 CALL GCHAR(A+AA,B+BB,TE)
1520 L=A
1530 LL=B
1540 IF TE=129 THEN 3230
1550 L=L+AA
1560 LL=LL+BB
1570 CALL GCHAR(L,LL,TE)
1580 IF TE<128 THEN 1550
1590 CALL HCHAR(A+AA,B+BB,G(1))
1600 IF TE>135 THEN 1950
1610 ON TE-127 GOTO 1620,1730,1840
1620 CALL SOUND(1000,-6,0)
1630 CALL HCHAR(L,LL,45)
1640 FOR PO=1 TO 5
1650 CALL COLOR(2,9,16)
1660 FOR PA=1 TO 30
1670 NEXT PA
1680 CALL COLOR(2,16,9)
1690 NEXT PO
1700 CALL HCHAR(L,LL,130)
1710 CALL HCHAR(A+AA,B+BB,127)
1720 GOTO 2080
1730 CALL SOUND(500,-6,0)
1740 CALL HCHAR(L-AA,LL-BB,45)
1750 FOR PO=1 TO 5
1760 CALL COLOR(2,9,16)
1770 FOR PA=1 TO 30
1780 NEXT PA
1790 CALL COLOR(2,16,9)
1800 NEXT PO
1810 CALL HCHAR(L-AA,LL-BB,127)
1820 CALL HCHAR(A+AA,B+BB,127)
1830 GOTO 2080
1840 CALL SOUND(1000,-6,3)
1850 CALL HCHAR(L,LL,45)
1860 FOR PO=1 TO 5
1870 CALL COLOR(2,9,16)
1880 FOR PA=1 TO 30
1890 NEXT PA
1900 CALL COLOR(2,16,9)
1910 NEXT PO
1920 CALL HCHAR(L,LL,127)
1930 CALL HCHAR(A+AA,B+BB,127)
1940 GOTO 2080
1950 CALL SOUND(3000,-6,0)
1960 XZ=XZ+1
1970 X=1
1980 CALL HCHAR(L,LL,45)
1990 FOR PA=1 TO 5
2000 FOR PO=1 TO 16
2010 CALL SCREEN(PO)
2020 NEXT PO
2030 NEXT PA
2040 CALL SCREEN(6)
2050 CALL HCHAR(A+AA,B+BB,127)
2060 CALL HCHAR(L,LL,46)
2070 GOTO 3350
2075 REM *************
2076 REM ACTION CHAR 2
2077 REM *************
2080 CALL KEY(2,ASA,POL)
2090 IF (POL=0)+(ASA>3)THEN 800
2100 IF ASA<>1 THEN 2200
2110 CALL GCHAR(D+DD,E+EE,TE)
2120 IF TE<128 THEN 2150
2130 CALL SOUND(200,1000,4)
2140 GOTO 800
2150 CALL HCHAR(D,E,127)
2160 D=D+DD
2170 E=E+EE
2180 CALL HCHAR(D,E,F)
2190 GOTO 800
2200 IF ASA=3 THEN 2780
2210 IF ASA=2 THEN 2230
2220 ON DD+2 GOTO 2240,2440,2580
2230 ON DD+2 GOTO 2250,2450,2590
2240 ON EE+2 GOTO 2600,2460,2260
2250 ON EE+2 GOTO 2260,2320,2380
2260 DD=-1
2270 EE=0
2280 G(2)=152
2290 F=144
2300 CALL HCHAR(D,E,F)
2310 GOTO 800
2320 DD=-1
2330 EE=1
2340 G(2)=156
2350 F=149
2360 CALL HCHAR(D,E,F)
2370 GOTO 800
2380 DD=0
2390 EE=1
2400 G(2)=155
2410 F=147
2420 CALL HCHAR(D,E,F)
2430 GOTO 800
2440 ON EE+2 GOTO 2660,2080,2320
2450 ON EE+2 GOTO 2460,800,2520
2460 DD=-1
2470 EE=-1
2480 G(2)=159
2490 F=148
2500 CALL HCHAR(D,E,F)
2510 GOTO 800
2520 DD=1
2530 EE=1
2540 G(2)=157
2550 F=151
2560 CALL HCHAR(D,E,F)
2570 GOTO 800
2580 ON EE+2 GOTO 2720,2520,2380
2590 ON EE+2 GOTO 2600,2660,2720
2600 DD=0
2610 EE=-1
2620 G(2)=153
2630 F=145
2640 CALL HCHAR(D,E,F)
2650 GOTO 800
2660 DD=1
2670 EE=-1
2680 G(2)=158
2690 F=150
2700 CALL HCHAR(D,E,F)
2710 GOTO 800
2720 DD=1
2730 EE=0
2740 G(2)=154
2750 F=146
2760 CALL HCHAR(D,E,F)
2770 GOTO 800
2780 CALL SOUND(300,-5,0)
2790 CALL GCHAR(D+DD,E+EE,TE)
2800 L=D
2810 LL=E
2820 IF TE=129 THEN 1950
2830 L=L+DD
2840 LL=LL+EE
2850 CALL GCHAR(L,LL,TE)
2860 IF TE<128 THEN 2830
2870 CALL HCHAR(D+DD,E+EE,G(2))
2880 IF TE>135 THEN 3230
2890 ON TE-127 GOTO 2900,3010,3120
2900 CALL SOUND(1000,-6,0)
2910 CALL HCHAR(L,LL,45)
2920 FOR PO=1 TO 5
2930 CALL COLOR(2,9,16)
2940 FOR PA=1 TO 30
2950 NEXT PA
2960 CALL COLOR(2,16,9)
2970 NEXT PO
2980 CALL HCHAR(L,LL,130)
2990 CALL HCHAR(D+DD,E+EE,127)
3000 GOTO 800
3010 CALL SOUND(500,-6,0)
3020 CALL HCHAR(L-DD,LL-EE,45)
3030 FOR PO=1 TO 5
3040 CALL COLOR(2,9,16)
```

```
3050 FOR PA=1 TO 30
3060 NEXT PA
3070 CALL COLOR(2,16,9)
3080 NEXT PO
3090 CALL HCHAR(L-DD,LL-EE,127)
3100 CALL HCHAR(D+DD,E+EE,127)
3110 GOTO 800
3120 CALL SOUND(1000,-6,3)
3130 CALL HCHAR(L,LL,45)
3140 FOR PO=1 TO 5
3150 CALL COLOR(2,9,16)
3160 FOR PA=1 TO 30
3170 NEXT PA
3180 CALL COLOR(2,16,9)
3190 NEXT PO
3200 CALL HCHAR(L,LL,127)
3210 CALL HCHAR(D+DD,E+EE,127)
3220 GOTO 800
3230 CALL SOUND(3000,-6,0)
3240 ZX=ZX+1
3250 XX=1
3260 CALL HCHAR(L,LL,45)
3270 FOR PA=1 TO 5
3280 FOR PO=1 TO 16
3290 CALL SCREEN(PO)
3300 NEXT PO
3310 NEXT PA
3320 CALL SCREEN(6)
3330 CALL HCHAR(D+DD,E+EE,127)
3340 CALL HCHAR(L,LL,46)
3345 REM *********
3346 REM RESULTATS
3347 REM *********
3350 CALL KEY(0,ASA,POL)
3360 IF POL=0 THEN 3350
3370 CALL CLEAR
3380 CALL COLOR(2,2,1)
3390 IF X=1 THEN 3530
3400 QW$="NOIRS"
3410 QZ$="BLANCS"
3420 XX=0
3430 PRINT "LES "&QW$&" SONT LES MAITRES
 DE LA VILLE!": :"LES "&QZ$&" VEULENT I
LS UNE   AUTRE CHANCE (O/N)?": :
3440 PRINT "TABLEAU DE CHASSE:":"********
**********": :"BLANC:": :"NOIR :": : :
:
3450 CALL HCHAR(18,9,144,XZ)
3460 CALL HCHAR(20,9,136,ZX)
3470 CALL KEY(0,ASA,POL)
3480 IF POL=0 THEN 3470
3490 IF ASA<>78 THEN 510
3500 CALL CLEAR
3510 PRINT "DANS CE CAS JE DOIS EN      C
ONCLURE QUE LES "&QZ$&" SONT DES POULES
 MOUILLES": : : : : :
3520 END
3530 QW$="BLANCS"
3540 QZ$="NOIRS"
3550 X=0
3560 GOTO 3430
3570 PRINT "COMBAT DE CHARS": :"OCTOBRE 1
983": :"99 MAGAZINE ET":"HOET JEAN-CLAU
DE": : : : : :
3580 RETURN
```

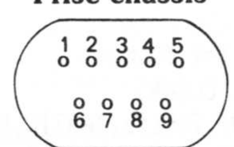
Réalisez votre cordon de magnétophone

Guy Fix

Vous êtes nombreux à ne pas pouvoir vous procurer le cordon qui permet de sauvegarder des programmes sur cassettes. Si le bricolage ne vous fait pas peur, les renseignements que nous allons vous donner pourrons vous être utiles.

Le raccordement au TI se fait par l'intermédiaire d'une prise CANON 9 points ou SOURIAU DEF-9S.

Prise chassis

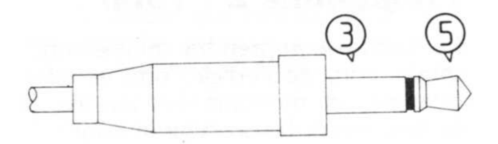


1 et 2 : télécommande moteur CS1

Fiche JACK 2,5 mm

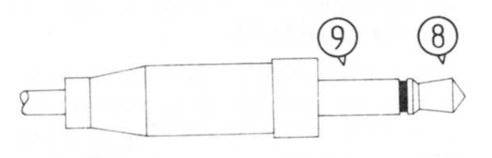
3 et 5 : sortie magnéto CS1

Fiche JACK 3,5 mm



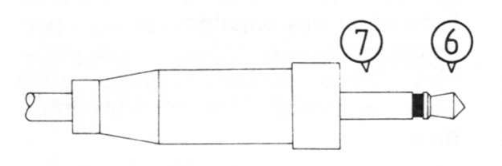
8 et 9 : entrée magnéto CS1

Fiche JACK 3,5 mm



6 et 7 : télécommande moteur CS2

Fiche JACK 2,5 mm



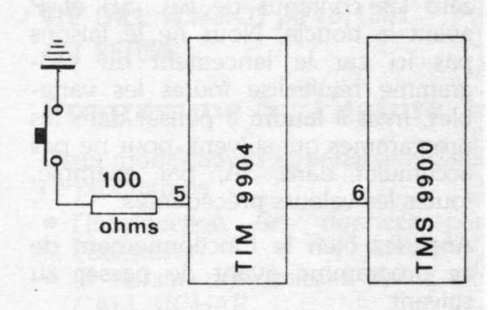
En cas de non-fonctionnement de la télécommande, inversez les fils au niveau de la fiche jack télécommande CS1 (ou CS2).

Pour le cordon magnétophone simple CS1, il ne faut pas brancher les fils télécommande moteur CS2 6 et 7.
La sortie magnétophone CS2 est en parallèle sur la sortie CS1 (3 et 5).
3 et 5 correspondent à la fiche du fil rouge (micro) du manuel TI.
8 et 9 correspondent à la fiche du fil blanc (écouteur ou haut parleur). ■

Une amélioration de la touche "RESET"

Robert Thibodeau - Québec

J'ai fait la modification pour le "RESET" et cela fonctionne parfaitement, sauf dans le cas où un synthétiseur de voix est raccordé au TI-99 car, avec cette configuration, il est impossible de reprendre le contrôle après avoir effectué un "RESET". Je me suis penché sur le problème et en suis arrivé à découvrir la solution suivante : au lieu de brancher le bouton sur la patte 6 du TMS 9900, il est possible de se placer sur la patte 5 du TIM 9904 situé juste au-dessus du TMS 9900, en ajoutant une résistance de 100 ohms.



Basic Etendu vers Basic TI

Lorsque l'on charge un programme écrit en Basic TI alors que l'on a choisi l'option 2 du menu principal, le programme ne fonctionne pas, du fait des différences entre les deux langages. Normalement, il faut à nouveau charger le programme, en Basic TI cette fois.

Nous pouvons éviter ces manipulations en utilisant la méthode suivante :
Entrez les lignes

```
1 !@P-
2 CALL COLOR
```

En faisant "RUN", vous obtenez un message d'erreur mais vous êtes passé en Basic TI. Il ne vous reste plus qu'à supprimer les lignes 1 et 2 et le programme fonctionnera normalement.

Apprendre le Basic en programmant

Roger Didi

Rappelons d'abord, pour le lecteur qui ne connaît pas notre méthode de travail, que nous construisons progressivement un programme en profitant des différentes étapes de cette construction pour préciser certaines notions.
Aujourd'hui, un client nous demande de lui construire un programme de facturation. Nous allons donc commencer par dessiner un tableau sur l'écran, puis nous apprendrons à y inscrire des données. Enfin, nous apprendrons à modifier ces données en fonction des décisions de l'utilisateur.
En ce qui concerne le Basic, nous profiterons de l'évolution de ce programme pour mieux comprendre l'utilisation des variables, et plus particulièrement des tableaux de variables. Nous perfectionnerons aussi notre approche des sous-programmes.

Programme 1 : Calculs

Ce premier programme est un essai qui permet d'observer les calculs utilisant des tableaux. A( ) et B( ) sont des tableaux à une ligne (ou une colonne) pouvant contenir jusqu'à 15 nombres (ligne 100).

SA contient la somme de tous les nombres contenus dans le tableau A( ).
SB contient la somme de tous les nombres contenus dans le tableau B( ).
P contient la somme de tous les produits des nombres de même numéro du tableau A( ) et du tableau B( ).

Normalement, il faudrait initialiser à zéro les contenus de SA, SB et P avant la boucle. Nous ne le faisons pas ici car le lancement du programme réinitialise toutes les variables, mais il faudra y penser dans les programmes qui suivent, pour ne pas accumuler dans SA, par exemple, toutes les valeurs précédentes.

Analysez bien le fonctionnement de ce programme avant de passer au suivant.

```
10 REM *****************
15 REM * PROGRAMME No 1 *
20 REM *    CALCULS     *
25 REM *****************
30 REM *   BASIC TI     *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM A(15),B(15)
110 FOR I=1 TO 15
120 INPUT A(I),B(I)
130 SA=SA+A(I)
140 SB=SB+B(I)
150 P=P+A(I)*B(I)
160 PRINT "SA = ";SA:"SB = ";SB:
"P = ";P
170 NEXT I
180 PRINT "SA = ";SA:"SB = ";SB:
"P = ";P
```

Programme 2 : Total

La facture comprendra quatre rubriques : nom de l'article, prix unitaire, quantité, et montant; en bas de la facture, nous devons voir le total.

Nous prévoyons donc quatre tableaux : ART$( ), PU$( ), QU$( ), et PR$( ). Chacun de ces tableaux peut contenir dix mots (ligne 120).
Nous créons d'autre part le caractère 123 qui permettra de dessiner des lignes verticales, et nous le plaçons dans la variable T$.

L'instruction VAL( ) permet de transformer en nombre la chaîne de caractères formée de chiffres, et écrite entre parenthèses. L'instruction STR$( ) accomplit l'action inverse.

Certains se demanderont pourquoi nous n'avons pas choisi directement des variables numériques pour enregistrer les nombres. L'une des raisons que nous pouvons invoquer est la crainte d'une erreur de manipulation de l'utilisateur. En effet, sur un INPUT X, par exemple, si l'utilisateur introduit autre chose qu'un chiffre, la machine affiche un message d'erreur (en anglais !) qui pourrait l'affoler... Une autre raison est l'homogénéité des variables, et nous en verrons l'intérêt plus loin.
Nous en trouvons d'ailleurs un exemple à la ligne 330 où nous voyons dans le même tableau une chaîne de caractères et des nombres.

En dehors de ces remarques, ce programme ne doit pas présenter trop de difficulté; faites-le fonctionner plusieurs fois et faites un bilan de tous ses défauts et des améliorations que vous envisagez, avant de passer au suivant.

```
10 REM *****************
15 REM * PROGRAMME No 2 *
20 REM *     TOTAL      *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 CALL CLEAR
110 CALL SCREEN(6)
120 DIM ART$(10),PU$(10),QU$(10)
,PR$(10)
130 ART$(0)="ARTICLE"
140 PU$(0)="P.U"
150 QU$(0)="QTE."
160 PR$(0)="A PAYER"
170 CALL CHAR(123,"1010101010101
010")
180 T$=CHR$(123)
190 INPUT "NOMBRE D'ARTICLES  ":
N
200 FOR I=1 TO N
210 CALL CLEAR
220 PRINT TAB(9);"ARTICLE No ";I
230 PRINT : : :
240 INPUT "DESIGNATION :":ART$(I
)
250 INPUT "PRIX UNITAIRE :":PU$(
I)
260 INPUT "QUANTITE :":QU$(I)
270 PRIX=VAL(PU$(I))*VAL(QU$(I))
280 TOTAL=TOTAL+PRIX
290 PR$(I)=STR$(PRIX)
300 NEXT I
310 CALL CLEAR
320 FOR I=0 TO N
330 PRINT TAB(2);ART$(I);TAB(9);
T$;TAB(10);PU$(I);TAB(15);T$;TAB
(16);QU$(I);TAB(
20);T$;TAB(21);PR$(I)
340 REM La tabulation permet une
    presentation en tableau
350 NEXT I
360 PRINT : :TAB(15);"TOTAL : ";
TOTAL
370 GOTO 370
380 REM
390 REM Pour arreter, faire FCTN
 4
400 REM
```

Programme 3 : Tableau

Nous nous intéressons dans ce programme à une amélioration de la présentation de la facture.
Les traits doivent être mieux dessinés et une flèche que nous pourrions commander avec les touches fléchées du clavier (FCTN S code 8, et FCTN D code 9) désignera la colonne dans laquelle il sera possible d'intervenir.

Nous créons donc le caractère 132 (ligne 190) qui est de couleur blanche (ligne 120).

Pour comprendre les calculs, il faut se souvenir de deux choses :

- Une relation entre deux grandeurs prend la valeur −1 lorsqu'elle est vraie et 0 (zéro) lorsqu'elle est fausse.
- Lors d'un test IF... THEN... , la machine teste la nullité de l'expression écrite entre IF et THEN. Si c'est nul, elle passe à la ligne suivante (ou au renvoi indiqué après ELSE), sinon elle saute au renvoi indiqué après le THEN.

Le sous-programme commençant à la ligne 320 efface donc la flèche située en XF, calcule la nouvelle valeur de XF suivant la touche pressée, et redessine la flèche au bon endroit.
Souvenons-nous que XF contiendra le numéro de la colonne où se trouve la flèche.

Comme pour le programme précédent, soyez exigeant et analysez toutes les déficiences de celui-ci.

```
10 REM ******************
15 REM * PROGRAMME No 3 *
20 REM *    TABLEAU     *
25 REM ******************
30 REM *    BASIC TI    *
35 REM ******************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM ******************
60 REM
100 CALL CLEAR
110 CALL SCREEN(6)
120 CALL COLOR(13,16,1)
130 DIM A(20),B(20),ART$(10),PU$
(10),QU$(10),PR$(10)
140 ART$(0)="ARTICLE"
150 PU$(0)="P.U"
160 QU$(0)="QTE."
170 PR$(0)="A PAYER"
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 T$=CHR$(123)
210 XF=7
220 PRINT TAB(2);ART$(0);TAB(11)
;PU$(0);TAB(16);QU$(0);TAB(21);P
R$(0)
230 FOR I=1 TO 21
240 PRINT TAB(1);T$;TAB(9);T$;TA
B(15);T$;TAB(20);T$;TAB(28);T$
250 NEXT I
260 GOSUB 320
270 CALL KEY(0,K,S)
280 IF S=0 THEN 270
290 IF (K<>9)*(K<>8)THEN 270
300 GOSUB 320
310 GOTO 270
320 CALL HCHAR(1,XF,32)
330 XF=XF-6*(K=9)+6*(K=8)
340 XF=-25*(XF>25)-7*(XF<7)+XF*(
XF>=7)*(XF<=25)
350 CALL HCHAR(1,XF,132)
360 RETURN
```

Programme 4 : Deux flèches

Notre client voudrait voir une seconde flèche qui indiquerait la ligne sur laquelle il pourrait écrire, et se déplacerait horizontalement conjointement avec la flèche du haut, et verticalement sous l'action des deux autres flèches du clavier (FCTN X code 10 et FCTN E code 11).

Cette seconde flèche est déterminée de la façon suivante :

- caractère 133 dessiné à la ligne 200;
- position horizontale : variable XF augmentée de 6. Il faut donc modifier la position des traits verticaux et créer entre eux un intervalle régulier de sept caractères;
- position verticale : la variable YF est calculée aux lignes 380 et 390 et est comprise entre 4 et 18. Elle augmente de 1 si K=10 et diminue de 1 si K=11.

La variable XF est calculée de telle façon que la flèche du haut désigne toujours le premier caractère de l'intitulé de la colonne.
Etudiez la ligne 310 où la multiplication permet de réaliser un "ET" logique. Il nous reste à écrire le contenu de la facture; c'est ce que nous allons entreprendre dans les programmes qui suivent.

```
10 REM ******************
15 REM * PROGRAMME No 4 *
20 REM *  DEUX FLECHES  *
25 REM ******************
30 REM *    BASIC TI    *
35 REM ******************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM ******************
60 REM
100 DIM A(20),B(20),ART$(10),PU$
(10),QU$(10),PR$(10)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0)=" ARTIC"
150 PU$(0)=" P.U."
160 QU$(0)=" QTE. "
170 PR$(0)=" PRIX"
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 T$=CHR$(123)
220 XF=4
230 YF=4
240 PRINT TAB(1);ART$(I);TAB(8);
PU$(I);TAB(15);QU$(I);TAB(22);PR
$(I)
250 FOR I=1 TO 21
260 PRINT TAB(1);T$;TAB(8);T$;TA
B(15);T$;TAB(22);T$
270 NEXT I
280 GOSUB 340
290 CALL KEY(0,K,S)
300 IF S=0 THEN 290
310 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)THEN 290
320 GOSUB 340
330 GOTO 290
340 CALL HCHAR(1,XF,32)
350 CALL HCHAR(YF,XF+6,123)
360 XF=XF-7*(K=9)+7*(K=8)
370 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF>=4)*(XF<=25)
380 YF=YF-(K=10)+(K=11)
390 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF>=4)*(YF<=18)
400 CALL HCHAR(1,XF,132)
410 CALL HCHAR(YF,XF+6,133)
420 RETURN
```

Programme 5 : Facture

Deux modifications apparaissent dans ce programme :

- l'introduction des données par l'utilisateur;
- le dessin des traits à l'aide de CALL VCHAR.

En ce qui concerne ce second point, il faut simplement remarquer que CALL VCHAR (comme CALL HCHAR) utilise 32 caractères par lignes, alors que TAB n'en utilise que 28. Le caractère 1 pour TAB devient donc le caractère 3 pour CALL VCHAR, et il devient possible de

dessiner le trait vertical à droite de la facture. Il faut d'autre part dessiner les traits après l'écriture du texte pour éviter l'effet de "scroll" (déroulement) qui ferait monter les lignes.

```
10 REM *****************
15 REM * PROGRAMME No 5 *
20 REM *    FACTURE     *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10),PU$(10),QU$(10)
,PR$(10)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0)=" ARTIC"
150 PU$(0)=" P.U."
160 QU$(0)=" QTE. "
170 PR$(0)=" PRIX"
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 T$=CHR$(123)
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 CALL CLEAR
250 PRINT TAB(9);"ARTICLE No ";I
260 PRINT : : :"DESIGNATION--> "
;
270 INPUT ART$(I)
280 INPUT "PRIX UNITAIRE -->":PU
$(I)
290 INPUT "QUANTITE --->":QU$(I)
300 PRIX=VAL(PU$(I))*VAL(QU$(I))
310 PR$(I)=STR$(PRIX)
320 NEXT I
330 CALL CLEAR
340 XF=4
350 YF=4
360 FOR I=0 TO NB
370 PRINT TAB(1);ART$(I);TAB(8);
PU$(I);TAB(15);QU$(I);TAB(22);PR
$(I)
380 NEXT I
390 FOR I=1 TO 20-NB
400 PRINT
410 NEXT I
420 FOR I=3 TO 32 STEP 7
430 CALL VCHAR(5,I,123,20)
440 NEXT I
450 GOSUB 510
460 CALL KEY(0,K,S)
470 IF S=0 THEN 460
480 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)THEN 460
490 GOSUB 510
500 GOTO 460
510 CALL HCHAR(1,XF,32)
520 CALL HCHAR(YF,XF+6,123)
530 XF=XF-7*(K=9)+7*(K=8)
540 XF=-25*(XF)25)-4*(XF(4)+XF*(
XF)=4)*(XF(=25)
550 YF=YF-(K=10)+(K=11)
560 YF=-18*(YF)18)-4*(YF(4)+YF*(
YF)=4)*(YF(=18)
570 CALL HCHAR(1,XF,132)
580 CALL HCHAR(YF,XF+6,133)
590 RETURN
600 L=LEN(MOT$)
610 FOR I=0 TO L-1
620 K=ASC(SEG$(MOT$,L-I,1))
630 CALL HCHAR(YF,XF+5-I,K)
640 NEXT I
650 RETURN
```

Programme 6 : Textes

Les traits dessinés par le programme précédent effacent le premier caractère de la première colonne; d'autre part, l'instruction PRINT provoquant un "scroll" de l'écran vers le haut, il sera difficile à l'utilisateur d'introduire un nouveau texte sans détruire le tableau.

Nous allons donc apporter deux améliorations :

- mieux structurer les variables utilisées pour le texte;
- créer une routine qui affiche un texte (de 6 caractères) en un point quelconque de l'écran.

Pour bien suivre le fonctionnement du programme (et pour bien suivre les exigences de l'utilisateur) nous avons représenté l'écran dans un rectangle de 24 lignes par 32 colonnes. Il est d'ailleurs conseillé d'avoir continuellement à sa disposition de tels modèles rectangulaires.

Nous allons remplacer les tableaux à un indice des programmes précédents par un tableau unique ART$(,) à deux indices.

Le premier indice est réservé au numéro de la colonne et prend donc les valeurs de 0 à 3.

Ainsi ART$(I,J) contiendra le texte qui doit être écrit à la Ième ligne de la colonne J. Attention, il s'agit ici de lignes et de colonnes de texte et non de celles de l'écran. Il faut donc trouver une relation entre I et J d'une part, et la position X,Y du texte sur l'écran.

Nous avons choisi de placer les intitulés des colonnes sur la deuxième ligne d'écran et la première ligne de la facture sur la sixième ligne d'écran. Un raisonnement simple permet alors de comprendre le but des lignes 380-390 et 480-490 du programme.

La routine commençant à la ligne 680 écrit un mot de six lettres nommé MOT$ en un point quelconque de l'écran : ligne X, colonne Y. Si le mot est formé de moins de six lettres, il est complété par des blancs : à droite pour le texte, à gauche pour les nombres (ce qui explique les lignes 690-720).

```
10 REM *****************
15 REM * PROGRAMME No 6 *
20 REM *     TEXTES     *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101
010")
190 CALL CHAR(132,"18181818FF7E3
C18")
200 CALL CHAR(133,"183878FFFF783
818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : "
:NB
230 FOR I=1 TO NB
240 CALL CLEAR
250 PRINT TAB(9);"ARTICLE ";I
260 PRINT : : :
270 INPUT "DESIGNATION --> ":ART
$(I,0)
280 INPUT "PRIX UNITAIRE -->":AR
T$(I,1)
290 INPUT "QUANTITE -->":ART$(I,
2)
300 PRIX=VAL(ART$(I,1))*VAL(ART$
(I,2))
310 ART$(I,3)=STR$(PRIX)
320 NEXT I
330 CALL CLEAR
340 XF=4
350 YF=4
360 FOR J=0 TO 3
```

```
370 MOT$=ART$(0,J)
380 X=J*7+3
390 Y=2
400 GOSUB 680
410 NEXT J
420 FOR I=3 TO 32 STEP 7
430 CALL VCHAR(3,I,123,20)
440 NEXT I
450 FOR I=1 TO NB
460 FOR J=0 TO 3
470 MOT$=ART$(I,J)
480 X=7*J+3
490 Y=5+I
500 GOSUB 680
510 NEXT J
520 NEXT I
530 GOSUB 590
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 IF (K<>9)*(K<>8)*(K<>11)*(K<>10)THEN 540
570 GOSUB 590
580 GOTO 540
590 CALL HCHAR(1,XF,32)
600 CALL HCHAR(YF,XF+6,123)
610 XF=XF-7*(K=9)+7*(K=8)
620 XF=-25*(XF>25)-4*(XF<4)+XF*(XF>=4)*(XF<=25)
630 YF=YF-(K=10)+(K=11)
640 YF=-18*(YF>18)-4*(YF<4)+YF*(YF>=4)*(YF<=18)
650 CALL HCHAR(1,XF,132)
660 CALL HCHAR(YF,XF+6,133)
670 RETURN
680 L=LEN(MOT$)
690 IF J<>0 THEN 720
700 MOT$=MOT$&SEG$(BL$,1,6-L)
710 GOTO 730
720 MOT$=SEG$(BL$,1,6-L)&MOT$
730 FOR U=1 TO 6
740 K=ASC(SEG$(MOT$,U,1))
750 CALL HCHAR(Y,X+U,K)
760 NEXT U
770 RETURN
```

Programme 7 : Total

Nous ajoutons dans ce programme une routine qui calcule le total et l'affiche à la ligne 20, colonne 24. Ce programme devrait être aisément compris et nous n'insistons pas. Etudiez-le calmement...

```
10 REM *****************
15 REM * PROGRAMME No 7 *
20 REM *     TOTAL      *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101010")
190 CALL CHAR(132,"18181818FF7E3C18")
200 CALL CHAR(133,"183878FFFF783818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : ":NB
230 FOR I=1 TO NB
240 PRINT : : :
250 PRINT TAB(9);"ARTICLE ";I
260 INPUT "DESIGNATION --> ":ART$(I,0)
270 INPUT "PRIX UNITAIRE -->":ART$(I,1)
280 INPUT "QUANTITE -->":ART$(I,2)
290 PRIX=VAL(ART$(I,1))*VAL(ART$(I,2))
300 ART$(I,3)=STR$(PRIX)
310 NEXT I
320 CALL CLEAR
330 XF=4
340 YF=4
350 FOR J=0 TO 3
360 MOT$=ART$(0,J)
370 X=J*7+3
380 Y=2
390 GOSUB 680
400 NEXT J
410 FOR I=3 TO 32 STEP 7
420 CALL VCHAR(3,I,123,20)
430 NEXT I
440 FOR I=1 TO NB
450 FOR J=0 TO 3
460 MOT$=ART$(I,J)
470 X=7*J+3
480 Y=5+I
490 GOSUB 680
500 NEXT J
510 NEXT I
520 GOSUB 780
530 GOSUB 590
540 CALL KEY(0,K,S)
550 IF S=0 THEN 540
560 IF (K<>9)*(K<>8)*(K<>11)*(K<>10)THEN 540
570 GOSUB 590
580 GOTO 540
590 CALL HCHAR(1,XF,32)
600 CALL HCHAR(YF,XF+6,123)
610 XF=XF-7*(K=9)+7*(K=8)
620 XF=-25*(XF>25)-4*(XF<4)+XF*(XF>=4)*(XF<=25)
630 YF=YF-(K=10)+(K=11)
640 YF=-18*(YF>18)-4*(YF<4)+YF*(YF>=4)*(YF<=18)
650 CALL HCHAR(1,XF,132)
660 CALL HCHAR(YF,XF+6,133)
670 RETURN
680 L=LEN(MOT$)
690 IF J<>0 THEN 720
700 MOT$=MOT$&SEG$(BL$,1,6-L)
710 GOTO 730
720 MOT$=SEG$(BL$,1,6-L)&MOT$
730 FOR U=1 TO 6
740 K=ASC(SEG$(MOT$,U,1))
750 CALL HCHAR(Y,X+U,K)
760 NEXT U
770 RETURN
780 FOR T=1 TO NB
790 TOTAL=TOTAL+VAL(ART$(T,3))
800 NEXT T
810 TOT$=STR$(TOTAL)
820 X=24
830 Y=20
840 MOT$=TOT$
850 J=1
860 GOTO 680
```

Programme 8 : Variations

L'utilisateur veut pouvoir modifier sa facture, changer les prix ou les quantités, ajouter des lignes : nous lui offrons la possibilité de le faire en actionnant FCTN 8 (REDO) dont le code est 6. Ceci explique les modifications de la ligne de test 570, et la ligne 580 qui renvoie à la ligne 900 si REDO est actionnée.
Il faut maintenant calculer les indices I et J dans le tableau ART$(,) en fonction de la position XF et YF des deux flèches (ce qui est fait simplement aux lignes 900 et 910) et la position du nouveau texte (lignes 920 et 930).

Si l'utilisateur veut ajouter un article, le nombre contenu dans la variable NB doit augmenter : 950-970.
On efface les anciens contenus : lignes 980-1000.
Nouvelle scrutation du clavier : lignes 1010-1020.

Lorsque l'utilisateur a terminé son nouveau texte, il doit presser la touche "ENTER" (code 13). Ceci lui est permis grâce à la ligne 1030 qui renvoie à 1080. La suite du sous-pro-

gramme est évidente. Remarquez seulement la réinitialisation de la variable TOTAL (qui doit être évidemment recalculée avec les nouvelles données).

```
10 REM *****************
15 REM * PROGRAMME No 8 *
20 REM *   VARIATIONS   *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101010")
190 CALL CHAR(132,"18181818FF7E3C18")
200 CALL CHAR(133,"183878FFFF783818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : ":NB
230 FOR I=1 TO NB
240 PRINT : : :
250 PRINT TAB(9);"ARTICLE ";I
260 INPUT "DESIGNATION --> ":ART$(I,0)
270 INPUT "PRIX UNITAIRE -->":ART$(I,1)
280 INPUT "QUANTITE -->":ART$(I,2)
290 PRIX=VAL(ART$(I,1))*VAL(ART$(I,2))
300 ART$(I,3)=STR$(PRIX)
310 NEXT I
320 CALL CLEAR
330 XF=4
340 YF=4
350 FOR J=0 TO 3
360 MOT$=ART$(0,J)
370 X=J*7+3
380 Y=2
390 GOSUB 710
400 NEXT J
410 FOR I=3 TO 32 STEP 7
420 CALL VCHAR(3,I,123,20)
430 NEXT I
440 FOR I=1 TO NB
450 FOR J=0 TO 3
460 MOT$=ART$(I,J)
470 X=7*J+3
480 Y=5+I
490 GOSUB 710
500 NEXT J
510 NEXT I
520 GOSUB 810
530 GOSUB 620
540 CALL SOUND(200,500,3,-4,5)
550 CALL KEY(0,K,S)
560 IF S=0 THEN 550
570 IF (K<>9)*(K<>8)*(K<>11)*(K<>10)*(K<>6)THEN 540
580 IF K=6 THEN 900
590 GOSUB 620
600 GOTO 540
620 CALL HCHAR(1,XF,32)
630 CALL HCHAR(YF,XF+6,123)
640 XF=XF-7*(K=9)+7*(K=8)
650 XF=-25*(XF>25)-4*(XF<4)+XF*(XF>=4)*(XF<=25)
660 YF=YF-(K=10)+(K=11)
670 YF=-18*(YF>18)-4*(YF<4)+YF*(YF>=4)*(YF<=18)
680 CALL HCHAR(1,XF,132)
690 CALL HCHAR(YF,XF+6,133)
700 RETURN
710 L=LEN(MOT$)
720 IF J<>0 THEN 750
730 MOT$=MOT$&SEG$(BL$,1,6-L)
740 GOTO 760
750 MOT$=SEG$(BL$,1,6-L)&MOT$
760 FOR U=1 TO 6
770 K=ASC(SEG$(MOT$,U,1))
780 CALL HCHAR(Y,X+U,K)
790 NEXT U
800 RETURN
810 FOR T=1 TO NB
820 TOTAL=TOTAL+VAL(ART$(T,3))
830 NEXT T
840 TOT$=STR$(TOTAL)
850 X=24
860 Y=20
870 MOT$=TOT$
880 J=1
890 GOTO 710
900 J=(XF-3)/7
910 I=YF-5
920 X=XF-1
930 Y=YF
940 IF I<=NB THEN 980
950 NB=I
960 ART$(I,1)="0"
970 ART$(I,2)="0"
980 MOT$=""
990 ART$(I,J)=""
1000 GOSUB 710
1010 CALL KEY(0,C,S)
1020 IF S=0 THEN 1010
1030 IF C=13 THEN 1080
1040 ART$(I,J)=ART$(I,J)&CHR$(C)
1050 MOT$=ART$(I,J)
1060 GOSUB 710
1070 GOTO 1010
1080 PRIX=VAL(ART$(I,1))*VAL(ART$(I,2))
1090 ART$(I,3)=STR$(PRIX)
1100 TOTAL=0
1110 GOTO 440
```

Programme 9 : Prix TTC

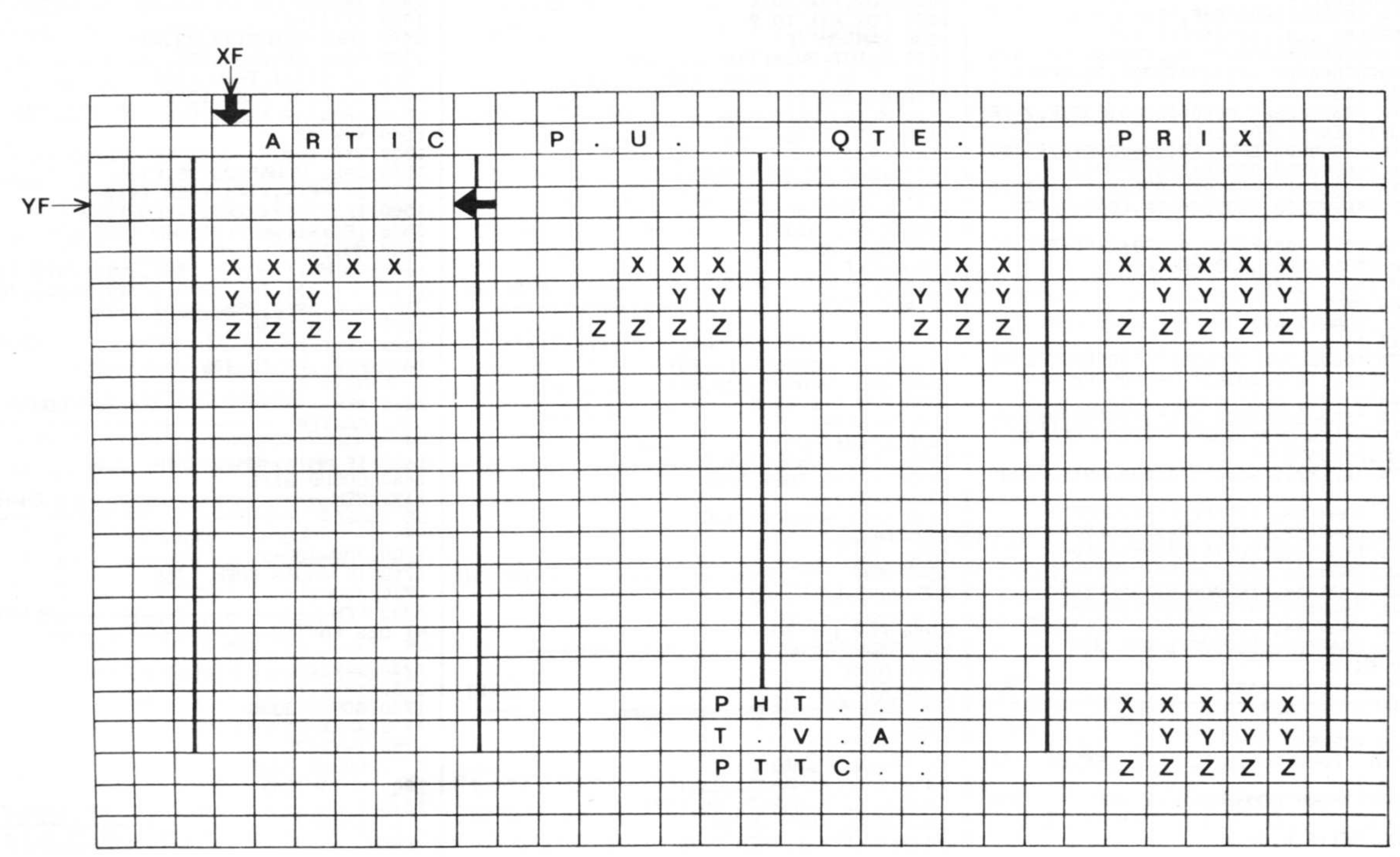
Nous laissons le soin au lecteur d'étudier ce programme qui apporte quelques modifications :

- quelques bips de tonalités différentes permettent d'éveiller l'attention de l'utilisateur ou de lui annoncer la fin d'une correction;
- la TVA et le prix TTC sont maintenant affichés et corrigés selon ses souhaits.

Ces programmes auraient besoin d'être encore améliorés. Si vous souhaitez nous proposer des modifications ou si vous désirez que nous vous en proposions, n'hésitez pas à nous écrire. ■

```
10 REM *****************
15 REM * PROGRAMME No 9 *
20 REM *    PRIX TTC    *
25 REM *****************
30 REM *    BASIC TI    *
35 REM *****************
40 REM *   COPYRIGHT    *
45 REM *   ROGER DIDI   *
50 REM * ET 99 MAGAZINE *
55 REM *****************
60 REM
100 DIM ART$(10,4)
110 CALL CLEAR
120 CALL COLOR(13,16,1)
130 CALL SCREEN(6)
140 ART$(0,0)=" ARTIC"
150 ART$(0,1)=" P.U. "
160 ART$(0,2)=" QTE. "
170 ART$(0,3)=" PRIX "
180 CALL CHAR(123,"1010101010101010")
190 CALL CHAR(132,"18181818FF7E3C18")
200 CALL CHAR(133,"183878FFFF783818")
210 BL$="      "
220 INPUT "NOMBRE D'ARTICLES : ":NB
230 FOR I=1 TO NB
240 PRINT : : :
250 PRINT TAB(9);"ARTICLE ";I
260 INPUT "DESIGNATION --> ":ART$(I,0)
270 INPUT "PRIX UNITAIRE -->":ART$(I,1)
280 INPUT "QUANTITE -->":ART$(I,2)
290 PRIX=VAL(ART$(I,1))*VAL(ART$(I,2))
300 ART$(I,3)=STR$(PRIX)
```

```
310 CALL CLEAR
320 NEXT I
330 CALL CLEAR
340 CALL SOUND(300,120,2)
350 INPUT "TAUX DE TVA --> ":TVA
$
360 CALL CLEAR
370 XF=4
380 YF=4
390 FOR J=0 TO 3
400 MOT$=ART$(0,J)
410 X=J*7+3
420 Y=2
430 GOSUB 860
440 NEXT J
450 FOR I=3 TO 32 STEP 7
460 CALL VCHAR(3,I,123,20)
470 NEXT I
480 MOT$="PHT..."
490 X=15
500 Y=20
510 J=0
520 GOSUB 860
530 MOT$="T.V.A."
540 Y=21
550 GOSUB 860
560 MOT$="PTTC.."
570 Y=22
580 GOSUB 860
590 FOR I=1 TO NB
600 FOR J=0 TO 3
610 MOT$=ART$(I,J)
620 X=7*J+3
630 Y=5+I
640 GOSUB 860
650 NEXT J
660 NEXT I
670 GOSUB 960
680 GOSUB 770
690 CALL SOUND(200,500,3,-4,5)
700 CALL KEY(0,K,S)
710 IF S=0 THEN 700
720 IF (K<>9)*(K<>8)*(K<>11)*(K<
>10)*(K<>6)THEN 690
730 IF K=6 THEN 1130
740 GOSUB 770
750 GOTO 690
770 CALL HCHAR(1,XF,32)
780 CALL HCHAR(YF,XF+6,123)
790 XF=XF-7*(K=9)+7*(K=8)
800 XF=-25*(XF>25)-4*(XF<4)+XF*(
XF)=4)*(XF<=25)
810 YF=YF-(K=10)+(K=11)
820 YF=-18*(YF>18)-4*(YF<4)+YF*(
YF)=4)*(YF<=18)
830 CALL HCHAR(1,XF,132)
840 CALL HCHAR(YF,XF+6,133)
850 RETURN
860 L=LEN(MOT$)
870 IF J<>0 THEN 900
880 MOT$=MOT$&SEG$(BL$,1,6-L)
890 GOTO 910
900 MOT$=SEG$(BL$,1,6-L)&MOT$
910 FOR U=1 TO 6
920 K=ASC(SEG$(MOT$,U,1))
930 CALL HCHAR(Y,X+U,K)
940 NEXT U
950 RETURN
960 FOR T=1 TO NB
970 TOTAL=TOTAL+VAL(ART$(T,3))
980 NEXT T
990 TOT$=STR$(TOTAL)
1000 X=24
1010 Y=20
1020 MOT$=TOT$
1030 J=1
1040 GOSUB 860
1050 TAXE=TOTAL*VAL(TVA$)/100
1060 MOT$=SEG$(STR$(TAXE),1,6)
1070 Y=21
1080 GOSUB 860
1090 PTTC=TOTAL+TAXE
1100 MOT$=SEG$(STR$(PTTC),1,6)
1110 Y=22
1120 GOTO 860
1130 J=(XF-3)/7
1140 I=YF-5
1150 X=XF-1
1160 Y=YF
1170 IF I<=NB THEN 1210
1180 NB=I
1190 ART$(I,1)="0"
1200 ART$(I,2)="0"
1210 MOT$=""
1220 ART$(I,J)=""
1230 GOSUB 860
1240 CALL KEY(0,C,S)
1250 IF S=0 THEN 1240
1260 IF C=13 THEN 1310
1270 ART$(I,J)=ART$(I,J)&CHR$(C)
1280 MOT$=ART$(I,J)
1290 GOSUB 860
1300 GOTO 1240
1310 PRIX=VAL(ART$(I,1))*VAL(ART
$(I,2))
1320 ART$(I,3)=STR$(PRIX)
1330 TOTAL=0
1340 GOTO 590
```



Jeu de mémorisation

Jean-Claude Cornevin

Le principe du jeu est simple : 18 paires de cartes sont retournées. Chaque joueur, à tour de rôle, retourne deux cartes.
Si les cartes sont identiques, elles sont retirées du jeu, le score du joueur est augmenté d'un point.
Si les cartes sont différentes, elles sont à nouveau retournées et le tour passe au joueur suivant.

Vous pourrez, avec ce jeu écrit en **Basic TI**, tester et exercer votre mémoire visuelle.

Les remarques insérées dans le listing vous permettront de comprendre le fonctionnement de ce programme. Que le meilleur gagne... ■




```
100 REM ******************
110 REM *     JEU  DE     *
120 REM *  MEMORISATION   *
130 REM ******************
140 REM *    BASIC  TI    *
150 REM ******************
160 REM *    COPYRIGHT    *
170 REM *  J.C. CORNEVIN  *
180 REM * ET 99 MAGAZINE *
190 REM ******************
200 CALL CLEAR
210 CALL SCREEN(2)
220 PRINT "                    JEU": : : : : :
230 PRINT "                     DE": : : : : :
240 PRINT "           MEMORISATION": : : :
250 FOR I=2 TO 9
260 CALL COLOR(I,16,1)
270 NEXT I
280 CALL COLOR(14,10,16)
290 GOSUB 3740
300 REM------------------------------ MOTI
F CARTES -----------------------------
---
310 FOR I=90 TO 156
320 READ A$
330 CALL CHAR(I,A$)
340 NEXT I
350 DATA 3F7F7F7F,000000000010101,000000
0000808080,0101010307070707,808080C0E0E
0E0E0,0F0701
360 DATA FFFFFFFFDFDF9F9,FFFFFF7F3F3F1F0
F,F1F1E1E1C1C181FF,070303010100000FF,00C
0F0FFFFFFFFFF
370 DATA 000001FFFFFFFFFF,FFFF3F0F010081F
3,F0E0C0F0FC3F7FFF,000000003F4F8787,000
00000FCF2E1E1
380 DATA 8787474F3FBF3F3F,E1E1E2F2FCFCFCF
C,3F3F1F1F0F03,FCFCF8F8F0C0,
390 DATA ,8855225588552255,00000000001C3E
7F,0000000000387CFE,7F7F7F3F1F0F0703
400 DATA FEFEFEFCF8F0E0C0,01,8,0018187E7E
1818,0000000000000001,0000000081020408,
0303030707 0F1F1F
410 DATA C0E0E0E0E0E0E0F,3F3F3F3F3F3F1C,F
8F8F8F8F8F0C,,,000000151F1F1F0F
420 DATA 00000054FCFCFCF8,070606060707070
7,F0F0F0F0F0F0707,070707070707,70F0F0F0
F,0100000103
430 DATA F860F0F8FC,00000000001F3F7F,7F7F
7F7F3F3F1F,E0F0F0F0F0E0C
440 DATA 00000000FCFCFCFC,,0000001818,000
0000101,000000808,00000107070F0F07
450 DATA 000080E0E0F0F0E,1B7C7FFFFF7F7D1B
,D8BE7EFFFFFEBED8,0307070F1F,C0E0E0F0F8
,,
460 DATA 00001F003FBF3F,00000606FEFEFE,0C
0C,FF087F53577F001,8000FFFCF0C
470 FOR I=33 TO 46
480 READ A$
490 CALL CHAR(I,A$)
500 NEXT I
510 DATA "0000202030F0D0F","000040C0C0C0
E0E0","1F3F7C1831010303","F0F0F8F8F8F8F
8F8","00","00","00"
520 DATA "0000000000616325","000000000086
C6A4","27371F0F0F2F373B","E4ECF8F0F0F4E
CDC","3D1F0E047C70"
530 DATA "BCF870203C0C","0000060810204080
"
540 CALL CHAR(58,"FFFFFFFFFFF8FFFF")
550 CALL CHAR(59,"FFFFFFFF9F4F0F9F")
560 CALL CHAR(60,"F8F1F2E4E4E2F1F0")
570 CALL CHAR(61,"7F3F3F1F1F1F0F8F")
580 CALL CHAR(62,"FCFAF6CEE9FDFFFF")
590 CALL CHAR(63,"4F2783E0FFFFFFFF")
600 CALL CHAR(64,"F0E080")
610 CALL CHAR(87,"0078787878787878")
620 DIM A(18)
630 REM-------------------------------PRES
ENTATION------------------------------
---
640 CALL CLEAR
650 PRINT "POUR CONSULTER LES REGLES   1":
 : : : : :
660 PRINT "POUR COMMENCER LE JEU      2":
 : : :
670 CALL KEY(0,K,S)
680 IF S<>1 THEN 670
690 IF K=49 THEN 3650
700 CALL CLEAR
710 JU=0
720 JO=0
730 JT=0
740 PRINT "NOM DU 1ER JOUEUR": : :
750 INPUT J1$
760 FOR I=1 TO 10
770 CALL GCHAR(23,I+4,PLAY(1,I))
780 NEXT I
790 CALL CLEAR
800 PRINT "NOM DU 2EME JOUEUR": : :
810 INPUT J2$
820 FOR I=1 TO 10
830 CALL GCHAR(23,I+4,PLAY(2,I))
840 NEXT I
850 GOSUB 3190
860 CALL CLEAR
870 CALL CHAR(81,"78F8F8F8F8")
880 REM------------------------------ MISE
 EN COULEUR---------------------------
---
890 CALL COLOR(1,2,1)
900 CALL COLOR(2,4,16)
910 FOR I=3 TO 8
920 CALL COLOR(I,2,16)
930 NEXT I
940 FOR I=1 TO 5
950 CALL COLOR(I+8,4+2*I,16)
960 NEXT I
970 CALL COLOR(14,10,16)
980 CALL COLOR(15,4,16)
990 CALL COLOR(16,2,6)
1000 CALL SCREEN(3)
1010 REM----------------------------------
CODAGE -------------------------------
----
1020 PRINT "             ooJOUEooXooooYooo"
1030 FOR I=1 TO 18
1040 A(I)=2
1050 NEXT I
1060 FOR Y=1 TO 4
1070 FOR X=1 TO 9
1080 RANDOMIZE
1090 Z=INT(RND*17)+1
1100 IF A(Z)<1 THEN 1140
1110 A(Z)=A(Z)-1
1120 ME(X,Y)=Z
1130 GOTO 1180
1140 Z=Z+1
1150 IF Z<19 THEN 1100
1160 Z=1
1170 GOTO 1100
1180 GOSUB 3330
1190 NEXT X
1200 NEXT Y
1210 REM------------------------ MISE EN
 PLACE TEXTE -------------------------
---
1220 CALL CHAR(159,"FFFFFFFFFFFFFFFF")
1230 CALL HCHAR(1,16,88)
1240 CALL VCHAR(12,2,89)
1250 FOR I=1 TO 9
1260 X=3+3*I
1270 C=48+I
1280 CALL HCHAR(3,X,C)
1290 IF I>4 THEN 1320
1300 Y=2+4*I
1310 CALL HCHAR(Y,4,C)
1320 NEXT I
1330 JOU=1
1340 REM------------------------- AFFICH
AGE JOUEUR --------------------------
---
1350 FOR I=1 TO 10
1360 CALL HCHAR(23,I+2,PLAY(JOU,I))
1370 NEXT I
1380 REM---------------------------- DEBUT
 DU JEU ------------------------------
---
1390 XX(2)=0
1400 FOR I=1 TO 2
1410 CALL HCHAR(23,23,159)
1420 CALL HCHAR(23,28,159)
1430 CALL KEY(0,K,ST)
1440 IF ST<>1 THEN 1430
1450 IF K>57 THEN 1430
1460 IF K<49 THEN 1430
1470 X=K-48
1480 XX(I)=X
1490 CALL VCHAR(23,23,K)
1500 CALL KEY(0,K,ST)
1510 IF ST<>1 THEN 1500
1520 IF K>52 THEN 1500
1530 IF K<49 THEN 1500
1540 Y=K-48
1550 YY(I)=Y
1560 CALL VCHAR(23,28,K)
1570 IF ME(X,Y)=0 THEN 1410
1580 IF XX(1)<>XX(2)THEN 1600
1590 IF YY(1)=YY(2)THEN 1410
1600 MM(I)=ME(X,Y)
1610 ON MM(I)GOSUB 1930,2000,2070,2140,22
10,2280,2350,2420,2490,2560,2630,2700,2
770,2840,2910,2980,3050,3120
1620 GOSUB 3330
1630 NEXT I
1640 FOR I=1 TO 400
1650 NEXT I
1660 REM-------------------- COMPARAISON
 DES CARTES --------------------------
---
1670 IF MM(1)=MM(2)THEN 1820
1680 GOSUB 3190
1690 REM---------------------- CHANGEM.
DE JOUEUR ----------------------------
---
1700 JOU=JOU+1
1710 IF JOU<3 THEN 1740
1720 JOU=1
1730 REM-------------------- RETOURNEME
NT DES CARTES------------------------
---
1740 X=XX(1)
1750 Y=YY(1)
1760 GOSUB 3330
1770 X=XX(2)
1780 Y=YY(2)
1790 GOSUB 3330
1800 GOTO 1340
1810 REM---------------------------- AUGM
. DU SCORE---------------------------
---
```

```
1820 IF JOU=1 THEN 1850
1830 JU=JU+1
1840 GOTO 1860
1850 JO=JO+1
1860 JT=JO+JU
1870 REM---------------------- EFFACEMEN
T DES CARTES---------------------------
---
1880 GOSUB 3260
1890 ME(XX(1),YY(1))=0
1900 ME(XX(2),YY(2))=0
1910 IF JT=18 THEN 3430
1920 GOTO 1740
1930 B=136
1940 C=46
1950 D=137
1960 E=138
1970 F=150
1980 G=150
1990 RETURN
2000 B=102
2010 C=103
2020 D=152
2030 E=153
2040 F=154
2050 G=103
2060 RETURN
2070 B=102
2080 C=103
2090 D=155
2100 E=156
2110 F=102
2120 G=103
2130 RETURN
2140 B=141
2150 C=141
2160 D=141
2170 E=141
2180 F=141
2190 G=141
2200 RETURN
2210 B=144
2220 C=145
2230 D=146
2240 E=147
2250 F=148
2260 G=149
2270 RETURN
2280 B=104
2290 C=105
2300 D=106
2310 E=107
2320 F=108
2330 G=109
2340 RETURN
2350 B=119
2360 C=119
2370 D=119
2380 E=119
2390 F=119
2400 G=119
2410 RETURN
2420 B=120
2430 C=46
2440 D=122
2450 E=123
2460 F=124
2470 G=125
2480 RETURN
2490 B=128
2500 C=129
2510 D=130
2520 E=131
2530 F=132
2540 G=133
2550 RETURN
2560 B=40
2570 C=41
2580 D=42
2590 E=43
2600 F=44
2610 G=45
2620 RETURN
2630 B=96
2640 C=97
2650 D=98
2660 E=99
2670 F=100
2680 G=101
2690 RETURN
2700 B=113
2710 C=114
2720 D=115
2730 E=116
2740 F=117
2750 G=118
2760 RETURN
2770 B=142
2780 C=143
2790 D=142
2800 E=143
2810 F=142
2820 G=143
2830 RETURN
2840 B=58
2850 C=59
2860 D=60
2870 E=61
2880 F=62
2890 G=63
2900 RETURN
2910 B=150
2920 C=139
2930 D=150
2940 E=87
2950 F=90
2960 G=81
2970 RETURN
2980 B=33
2990 C=34
3000 D=35
3010 E=36
3020 F=134
3030 G=135
3040 RETURN
3050 B=91
3060 C=92
3070 D=93
3080 E=94
3090 F=95
3100 G=64
3110 RETURN
3120 B=141
3130 C=141
3140 D=142
3150 E=143
3160 F=141
3170 G=141
3180 RETURN
3190 B=112
3200 C=112
3210 D=112
3220 E=112
3230 F=112
3240 G=112
3250 RETURN
3260 B=37
3270 C=37
3280 D=37
3290 E=37
3300 F=37
3310 G=37
3320 RETURN
3330 REM-----------------------AFFICHAGE
 CARTES -------------------------------
---
3340 X1=3+3*X
3350 Y1=1+4*Y
3360 CALL HCHAR(Y1,X1,B)
3370 CALL HCHAR(Y1,X1+1,C)
3380 CALL HCHAR(Y1+1,X1,D)
3390 CALL HCHAR(Y1+1,X1+1,E)
3400 CALL HCHAR(Y1+2,X1,F)
3410 CALL HCHAR(Y1+2,X1+1,G)
3420 RETURN
3430 CALL CLEAR
3440 REM----------------------- AFFICHA
GE SCORE ------------------------------
---
3450 CALL SCREEN(16)
3460 IF JO<>JU THEN 3490
3470 PRINT "EGALITE 9 A 9": : :
3480 GOTO 3580
3490 PRINT "AU SCORE DE ";JO;" A ";JU: :
:
3500 PRINT "                GAGNE": : :
3510 IF JO>JU THEN 3540
3520 Q=2
3530 GOTO 3550
3540 Q=1
3550 FOR I=1 TO 10
3560 CALL HCHAR(21,I+2,PLAY(Q,I))
3570 NEXT I
3580 GOSUB 3740
3590 PRINT "POUR REJOUER      1": :
3600 PRINT "POUR ARRETER      2": :
3610 INPUT R
3620 IF R=1 THEN 700
3630 END
3640 REM------------------------ AFFICH.
REGLES --------------------------------
---
3650 CALL CLEAR
3660 PRINT "                REGLES": : :
3670 PRINT "IL S'AGIT D'UN JEU  FAISANT":
 :"APPEL A LA MEMOIRE": :"CHAQUE JOUEUR
 A TOUR DE ROLE": :
3680 PRINT "RETOURNE 2 CARTES": :"SI LES
DEUX IMAGES SONT IDEN": :"TIQUES  LE  J
OUEUR GAGNE  UN": :
3690 PRINT "POINT ET CONTINUE A JOUER": :
"SI LES IMAGES  SONT DIFFEREN": :"TES L
ES CARTES SE RETOURNENT": :
3700 PRINT "ET  LE  TOUR PASSE A L AUTRE"
: :"JOUEUR": :"APPUYER ENTER POUR CONTI
NUER"
3710 CALL KEY(0,K,S)
3720 IF S<>1 THEN 3710
3730 GOTO 700
3740 REM--------------------------SONORIS
ATION----------------------------------
---
3750 CALL SOUND(200,330,5)
3760 CALL SOUND(200,330,5)
3770 CALL SOUND(200,349,5)
3780 CALL SOUND(200,393,5)
3790 CALL SOUND(200,330,5)
3800 CALL SOUND(200,330,5)
3810 CALL SOUND(400,330,5)
3820 CALL SOUND(200,294,5)
3830 CALL SOUND(200,247,5)
3840 CALL SOUND(200,349,5)
3850 CALL SOUND(200,392,5)
3860 CALL SOUND(200,440,5)
3870 CALL SOUND(200,392,5)
3880 CALL SOUND(200,349,5)
3890 CALL SOUND(200,330,5)
3900 CALL SOUND(200,294,7)
3910 CALL SOUND(200,247,5)
3920 CALL SOUND(200,262,7)
3930 CALL SOUND(200,294,5)
3940 CALL SOUND(400,330,6)
3950 RETURN
```

Jack-Pot

Marianne Sutz

Ce programme, écrit en **Basic Etendu**, est très largement inspiré du célèbre jeu de casino du même nom. Dans un premier temps, le programme vous demande de rentrer la somme que vous désirez miser. La mise doit être inférieure à 10000.

Vous devez ensuite choisir les symboles à modifier. Par exemple, si vous frappez les touches 4 et 2 (l'ordre n'a pas d'importance), les symboles correspondant aux positions 2 et 4 seront changés. Si vous faites une erreur, vous pouvez tout annuler avec la touche 0. Une fois votre choix déterminé, vous devrez le valider avec la barre d'espacement.

La partie s'arrête lorsque vous obtenez quatre symboles identiques ou si votre score est inférieur ou égal à zéro. Avant de vous indiquer comment s'effectue le calcul des points en fonction des symboles visualisés, il faut signaler que l'apparition des flèches provoque un retirage aléatoire sur les quatre positions. Points obtenus (ou perdus) :

- quatre symboles quelconques identiques : +50 - fin de partie;
- quatre "dollars" : +100 - fin de partie;
- trois symboles quelconques identiques : +10;
- trois symboles quelconques identiques plus "dollar" : +20;
- trois "dollars" : +25;
- deux fois deux symboles quelconques identiques : score identique;
- deux symboles quelconques identiques : −5;
- couleurs (trèfle, coeur, carreau, pique) : +100;
- tous les symboles différents : −25%.

En fin de partie, le programme vous demande si vous désirez rejouer. Si vous répondez par la négative, une récapitulation des gains et pertes sera affichée. Une pression sur n'importe quelle touche interrompra définitivement le programme. Bonne chance...

■

```
100 ! ***********************
110 ! *     Jack - Pot      *
120 ! ***********************
130 ! *     Copyright       *
140 ! *   Marianne Sutz     *
150 ! * et "99 Magazine"    *
160 ! ***********************
170 ! *    Basic Etendu     *
180 ! ***********************
190 !
200 E$="3333F3F30303FFFF" :: F$="FFFFFFFF
FFFFFEFE" :: G$="FFFEFCF8F0E0C080" :: H
$="0103070F1F3F7FFF" :: Q$="FF7F3F1F0F070
301"
210 S$="FFFFC0C0CFCFCCCC" :: T$=RPT$("C",
16):: W$="FFFF0000FFFF0000" :: U$="CCCC
CFCFC0C0FFFF" :: Y$="0000FFFF0000FFFF" ::
 X$="FFFF0303F3F33333"
220 Z$=RPT$("3",16):: E$="3333F3F30303FFF
F"
230 R$="80C0E0F0F8FCFEFF" :: V$=RPT$("0",
16):: P$=RPT$("F",16):: OPTION BASE 1 :
: DIM GR$(10,4),C(10,2),SC(20,3)
240 C(1,1),C(1,2)=9 :: C(2,1),C(2,2),C(5,
2)=3 :: C(3,1),C(3,2)=14 :: C(4,1),C(4,
2),C(5,1)=10
```

```
250 C(6,1),C(6,2)=12 :: C(7,1),C(7,2)=13
:: C(8,1),C(8,2)=16 :: C(9,1),C(9,2)=2
:: C(10,1),C(10,2)=11
260 GR$(1,1)="070F1F3F3F7F7FFFFFFFFFFFFF
F7F7FC0F0F8FCFCFEFEFE"&P$ :: GR$(1,3)="
030F1F3F3F7F7F7F"&P$&"E0F0F8FCFCFEFEFF"&F
$
270 GR$(1,2)="3F3F1F1F0F0703010100000000
00000"&P$&"FFFF7F3F1F070301" :: GR$(1,4
)=P$&"FFFFFEFCF8E0C080FCFCF8F8F0E0C08080"
280 GR$(2,1)=V$&"00000F3F7F7FFFFF030F1F1F
3F3F3F3F1F1F0FCFFFFFFFFF" :: GR$(2,2)="
FFFF7F7F3F0F0000"&V$&"FFFFFFEFC707070707070
70F0F0F1F1F3F"
290 GR$(2,3)="C0F0F8F8FCFCFCFCF8F8F0F3FFF
FFFFF"&V$&"0000F0FCFEFEFFFF" :: GR$(2,4
)="FFFFFFF7E3E0E0E0E0E0F0F0F0F8F8FCFFFFFE
FEFCF0000000"
300 GR$(3,1)=V$&"0000010307070F0F01010103
070F1F3F7FFFFFFFFFFFFFFF" :: GR$(3,2)="
1F1F1F1F1F1F1F0F0F0701000000000000"&P$&"FFF
BE3030307070F"
310 GR$(3,3)="808080C0E0F0F8FCFEFFFFFFFFF
FFFFF"&V$&"000080C0E0E0F0F0" :: GR$(3,4
)=P$&"FFDFC7C0C0E0E0F0F8F8F8F8F8F8F0F0E
080"
320 GR$(4,2)=Q$&V$&P$&Q$ :: GR$(4,3)=R$&P
$&V$&R$ :: GR$(4,4)=P$&G$&G$&V$ :: GR$(
5,2)="FFFFFFFFFFFF7F7F7F3F3F1F1F0F0703"&P
$&"FFFFFFFFFEFCF8F0"
330 GR$(5,1)="0000000000000001070F1F3F3F7
F7FFF00000000000000F8FCFEFFFFFFFFFFFF"
:: GR$(5,4)="C0E0E0E0E0C0C0C080800000000
0000"&V$&V$
340 GR$(5,3)="000000103070E0C18383060600C
0C080387CFEC68703030300" :: GR$(6,1)="0
000000000000103070F0F1F3F7FFFFF000000073F
FFFFFF"&P$
350 GR$(6,2)="FFFF7F3F1F0F0F0703010000000
00000"&P$&"FFFFFF3F07" :: GR$(6,3)="000
000E0FCFFFFFF"&P$&"000000000000080C0E0F0F0
F8FCFEFFFF"
360 GR$(6,4)=P$&"FFFFFFFCE0000000FFFFFEFC
F8F0F0E0C080" :: GR$(7,1)=V$&"00070F1F1
F1F0F7F000000000000001071FFFFFFFFFFFFFF"
370 GR$(7,2)="FF8F1F1F1F0F0700"&V$&"FFFFF
FFFFFFFFF1F0701" :: GR$(7,3)="0000071F3
FFFFFFF"&P$&"0000C0F8FEFEFFFF"&F$
380 GR$(7,4)=P$&"FFFFFF3F1F070000FEFEFFFF
FFFFFFFFFFFFFFFEFEF8C0" :: GR$(8,1)=S$&T$
&W$&S$ :: GR$(8,2)=T$&U$&U$&Y$ :: GR$(8,3
)=W$&X$&X$&Z$
390 GR$(8,4)=E$&Y$&Z$&E$ :: GR$(9,1)=V$&"
0002060E1F3760C00103060C183C7C0C0C0C0C0
CFCFC"
400 GR$(9,2)="C060371F0E060200"&V$&"0000F
CFC0C0C0C0C0C7C3C180C060301" :: GR$(9,3
)="80C06030183C3E30303030303F3F0000"&V$&"
00406070F8EC0603"
410 GR$(9,4)="00003F3F30303030303E3C18306
0C0800306ECF87060400000"
420 GR$(10,1),GR$(10,3)="0202020F1F3F7A72
72727272727A3F1F404040F0F8FC5E4E4E40404
04040F0F8"
430 GR$(10,2),GR$(10,4)="0F02020202020272
727A3F1F0F020202FC5E4E4E4E4E4E4E4E5EFCF
8F0404040" :: GR$(4,1)=V$&H$&H$&P$
440 CALL CLEAR :: CALL SCREEN(5):: CALL M
AGNIFY(3):: CALL COLOR(2,10,2):: B2=48
:: CALL VCHAR(1,9,42,5):: CALL HCHAR(24,4
,42,4)
450 FOR B=1 TO 3 :: CALL VCHAR(1,B,42,24)
:: CALL HCHAR(B,4,42,4):: NEXT B :: FOR
 B=8 TO 10 STEP 2 :: CALL VCHAR(1,B,42,24
):: NEXT B
460 FOR B=6 TO 21 STEP 5 :: CALL VCHAR(B+
2,9,42,4):: CALL HCHAR(B+2,4,42,4):: B2
=B2+1 :: CALL VCHAR(B,9,B2):: NEXT B
470 CALL CHAR(136,GR$(10,1),140,GR$(10,2)
,60,GR$(10,3),44,GR$(10,4))
480 CALL CHAR(88,GR$(8,1),92,GR$(8,2),96,
GR$(8,3),100,GR$(8,4))
490 CALL CHAR(104,GR$(2,1),108,GR$(2,2),1
12,GR$(2,3),116,GR$(2,4))
500 CALL CHAR(120,GR$(1,1),124,GR$(1,2),1
28,GR$(1,3),132,GR$(1,4))
510 CALL SPRITE(#1,136,11,25,25,#2,140,11
,41,25,#3,60,11,25,41,#4,44,11,41,41)
520 CALL SPRITE(#5,88,16,65,25,#6,92,16,8
1,25,#7,96,16,65,41,#8,100,16,81,41)
530 CALL SPRITE(#9,104,3,105,25,#10,108,3
,121,25,#11,112,3,105,41,#12,116,3,121,
41)
540 CALL SPRITE(#13,120,9,145,25,#14,124,
9,161,25,#15,128,9,145,41,#16,132,9,161
,41)
550 FOR B=1 TO 24 :: DISPLAY AT(B,11)BEEP
:"$$$ JACK POT $$$" :: NEXT B :: FOR D=
0 TO 200 :: NEXT D :: GOSUB 1300
560 ON WARNING NEXT :: DISPLAY AT(3,10):"
VOTRE MISE SVP" :: ACCEPT AT(3,25)SIZE(
4)VALIDATE(DIGIT)BEEP:MI :: DISPLAY AT(3,
10):""
570 DISPLAY AT(3,11):"MISE";MI;"FRANCS" :
: CALL HCHAR(5,13,42,15):: CALL HCHAR(1
1,13,42,15):: CALL VCHAR(6,13,42,5):: CAL
L VCHAR(6,27,42,5)
580 DISPLAY AT(7,15)SIZE(8):"CREDIT:" ::
DISPLAY AT(8,12)SIZE(12):MI;"FRANCS"
590 XX=XX+1 :: SC(XX,1),CR=MI :: DISPLAY
AT(20,10):"TAPER LES NUMEROS" :: DISPLA
Y AT(21,13):"DE DESSIN A" :: DISPLAY AT(2
2,13):"REMPLACER ET"
600 DISPLAY AT(23,14):"FRAPPER LA" :: DIS
PLAY AT(24,10):"BARRE D'ESPACEMENT" ::
X1=10 :: X2=8 :: X3=2 :: X4=1
610 DA,DB,DC,DD=0
620 CALL KEY(0,R1,ST):: IF R1=32 THEN 680
 ELSE IF R1=48 THEN GOSUB 1310 :: GOTO
610 ELSE IF R1<49 OR R1>52 OR ST=0 OR ST=
-1 THEN 620
630 IF R1=49 THEN GOSUB 1280 :: DA=1 :: C
ALL HCHAR(7,9,43)
640 IF R1=50 THEN GOSUB 1280 :: DB=2 :: C
ALL HCHAR(12,9,43)
650 IF R1=51 THEN GOSUB 1280 :: DC=4 :: C
ALL HCHAR(17,9,43)
660 IF R1=52 THEN GOSUB 1280 :: DD=8 :: C
ALL HCHAR(22,9,43)
670 GOTO 620
680 X=DA+DB+DC+DD :: IF X=0 THEN 620
690 FOR B=0 TO 30 :: CALL SOUND(50,440,B,
-4,B):: NEXT B :: ON X GOTO 700,730,700
,760,700,730,700,790,700,730,700,760,700,
730,700
700 GOSUB 1270 :: X1=CA :: CALL CHAR(136,
GR$(CA,1),140,GR$(CA,2),60,GR$(CA,3),44
,GR$(CA,4))
710 CALL PATTERN(#1,136,#2,140,#3,60,#4,4
4):: CALL COLOR(#1,C(CA,1),#2,C(CA,1),#
3,C(CA,2),#4,C(CA,1))
720 GOSUB 1290 :: IF X=5 OR X=13 THEN 760
 ELSE IF X=9 THEN 790 ELSE IF X=1 THEN
810
730 GOSUB 1270 :: X2=CA :: CALL CHAR(88,G
R$(CA,1),92,GR$(CA,2),96,GR$(CA,3),100,
GR$(CA,4))
740 CALL PATTERN(#5,88,#6,92,#7,96,#8,100
```

```
):: CALL COLOR(#5,C(CA,1),#6,C(CA,1),#7
,C(CA,2),#8,C(CA,1))
750 GOSUB 1290 :: IF X=10 OR X=11 THEN 79
0 ELSE IF X=2 OR X=3 THEN 810
760 GOSUB 1270 :: X3=CA :: CALL CHAR(104,
GR$(CA,1),108,GR$(CA,2),112,GR$(CA,3),1
16,GR$(CA,4))
770 CALL PATTERN(#9,104,#10,108,#11,112,#
12,116):: CALL COLOR(#9,C(CA,1),#10,C(C
A,1),#11,C(CA,2),#12,C(CA,1))
780 GOSUB 1290 :: IF X=4 OR X=5 OR X=6 OR
 X=7 THEN 810
790 GOSUB 1270 :: X4=CA :: CALL CHAR(120,
GR$(CA,1),124,GR$(CA,2),128,GR$(CA,3),1
32,GR$(CA,4))
800 CALL PATTERN(#13,120,#14,124,#15,128,
#16,132):: CALL COLOR(#13,C(CA,1),#14,C
(CA,1),#15,C(CA,2),#16,C(CA,1)):: GOSUB 1
290
810 GOSUB 1310 :: IF X1=X2 AND X2=X3 AND
X3=X4 THEN IF X1=10 THEN CR=CR+100 :: G
OTO 940 ELSE CR=CR+50 :: GOTO 940
820 IF X1=X2 AND X2=X3 THEN IF X1=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X4=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
830 IF X1=X2 AND X2=X4 THEN IF X1=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X3=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
840 IF X2=X3 AND X3=X4 THEN IF X2=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X1=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
850 IF X1=X3 AND X3=X4 THEN IF X1=10 THEN
 CR=CR+20 :: GOTO 950 ELSE IF X2=10 THE
N CR=CR+25 :: GOTO 950 ELSE CR=CR+10 :: G
OTO 950
860 IF (X1=X2 AND X3=X4)OR(X1=X3 AND X2=X
4)OR(X1=X4 AND X2=X3)THEN 950
870 IF X1=X2 OR X1=X3 OR X1=X4 OR X2=X3 O
R X2=X4 OR X3=X4 THEN CR=CR-5 :: GOTO 9
50
880 IF X1+X2+X3+X4<>10 THEN 930
890 FOR B=5 TO 7 :: IF X1=B OR X2=B OR X3
=B OR X4=B THEN 930
900 NEXT B
910 DISPLAY AT(17,10):"**** COULEURS ****
" :: FOR ZA=110 TO 710 STEP 200 :: FOR
ZB=ZA TO ZA+200 STEP 20 :: CALL SOUND(-1,
ZB,0):: NEXT ZB
920 FOR ZC=ZA+100 TO ZA+300 STEP 20 :: CA
LL SOUND(-1,ZC,0):: NEXT ZC :: NEXT ZA
:: DISPLAY AT(17,10):"" :: CR=CR+100 :: G
OTO 950
930 CR=CR*.75 :: GOTO 950
940 DR=1
950 IF INT(CR)=0 THEN DISPLAY AT(8,12)SIZ
E(13):"" :: GOTO 970
960 DISPLAY AT(8,13)SIZE(5):USING "####":
INT(CR):: IF INT(CR)=1 THEN DISPLAY AT(
8,18)SIZE(7):"FRANC" ELSE DISPLAY AT(8,18
)SIZE(7):"FRANCS"
970 CRC=INT((CR-INT(CR))*100):: IF CRC=0
THEN DISPLAY AT(9,12)SIZE(13):"" :: GOT
O 990
980 DISPLAY AT(9,13)SIZE(3):USING "##":CR
C :: IF CRC=1 THEN DISPLAY AT(9,16)SIZE
(9):"CENTIME" ELSE DISPLAY AT(9,16)SIZE(9
):"CENTIMES"
990 IF CR<=0 OR DR=1 THEN 1030
1000 IF X1=9 OR X2=9 OR X3=9 OR X4=9 THEN
 1010 ELSE 610
1010 DISPLAY AT(13,14):"FLECHES !!" :: DI
SPLAY AT(14,12):"JE RELANCE LE" :: DISP
LAY AT(15,17):"JEU"
1020 FOR B=110 TO 1000 STEP 10 :: CALL SO
UND(-10,B,0):: NEXT B :: DISPLAY AT(13,
14):"" :: DISPLAY AT(14,12):"" :: DISPLAY
 AT(15,17):"" :: X=0 :: GOTO 700
1030 FOR B=0 TO 100 :: NEXT B :: FOR B=5
TO 24 :: DISPLAY AT(B,10)BEEP:"" :: NEX
T B :: CALL SCREEN(8):: DISPLAY AT(5,12):
"LA PARTIE EST"
1040 DISPLAY AT(6,14):"TERMINEE !"
1050 IF CR>MI THEN CRF=CR-MI :: SC(XX,2)=
INT(CR-MI):: DISPLAY AT(8,16):"GAIN:"
1060 IF CR<MI THEN CRF=MI-CR :: SC(XX,3)=
INT(MI-CR):: DISPLAY AT(8,16):"PERTE:"
1070 IF CR=MI THEN DISPLAY AT(8,12):"PART
IE NULLE!" :: GOTO 1150
1080 IF INT(CRF)=0 THEN 1090 ELSE IF INT(
CRF)=1 THEN DISPLAY AT(9,16):"1 FRANC"
ELSE DISPLAY AT(9,13):INT(CRF);"FRANCS"
1090 CRFC=INT((CRF-INT(CRF))*100)
1100 IF CRFC=0 THEN 1110 ELSE IF CRFC=1 T
HEN DISPLAY AT(10,14):"1 CENTIME" ELSE
DISPLAY AT(10,12):CRFC;"CENTIMES"
1110 IF CR>=0 THEN 1150 ELSE DISPLAY AT(1
2,11):"IL ME MANQUE DONC"
1120 IF INT(ABS(CR))=0 THEN 1130 ELSE IF
INT(ABS(CR))=1 THEN DISPLAY AT(13,15):"
1 FRANC" ELSE DISPLAY AT(13,13):INT(ABS(C
R));"FRANCS"
1130 CRFD=INT((ABS(CR)-INT(ABS(CR)))*100)
1140 IF CRFD=0 THEN 1150 ELSE IF CRFD=1 T
HEN DISPLAY AT(14,14):"1 CENTIME !" ELS
E DISPLAY AT(14,12):CRFD;"CENTIMES !"
1150 IF XX=20 THEN 1180 ELSE DISPLAY AT(1
8,11):"UNE AUTRE PARTIE" :: DISPLAY AT(
19,13):"OUI OU NON"
1160 CALL KEY(0,KK,ST):: IF ST=0 OR KK<78
 OR KK>79 THEN 1160
1170 IF KK=79 THEN DR=0 :: GOTO 440 ELSE
1190
1180 FOR B=0 TO 3000 :: NEXT B
1190 CALL DELSPRITE(ALL):: CALL CLEAR ::
CALL CHARSET :: CALL SCREEN(7):: FOR B=
5 TO 7 :: CALL COLOR(B,5,14):: NEXT B ::
DISPLAY AT(1,1):"PARTIE   MISE   GAIN
 PERTE"
1200 IMAGE ##_____####___####___####
1210 FOR B=1 TO XX :: DISPLAY AT(B+1,3):U
SING 1200:B,SC(B,1),SC(B,2),SC(B,3):: N
EXT B :: DISPLAY AT(23,1):"BILAN: MISE
 GAIN    PERTE"
1220 FOR B=1 TO XX :: SC1=SC1+SC(B,1):: S
C2=SC2+SC(B,2):: SC3=SC3+SC(B,3):: NEXT
 B
1230 IF SC2>SC3 THEN GAIN=SC2-SC3 :: PERT
E=0 ELSE IF SC2<SC3 THEN GAIN=0 :: PERT
E=SC3-SC2 ELSE GAIN,PERTE=0
1240 IMAGE ######  ######  ######
1250 DISPLAY AT(24,7):USING 1240:SC1,GAIN
,PERTE
1260 CALL KEY(0,KEY,ST):: IF ST=0 THEN 12
60 ELSE CALL CLEAR :: END
1270 RANDOMIZE :: CA=INT(RND*10)+1 :: RET
URN
1280 SON=(R1-47)*55 :: CALL SOUND(300,SON
,0,SON+50,1,SON+100,2):: RETURN
1290 CALL SOUND(100,400+(CA*50),0):: RETU
RN
1300 FOR B=2 TO 24 :: DISPLAY AT(B,9):""
:: NEXT B :: RETURN
1310 FOR B=7 TO 22 STEP 5 :: CALL HCHAR(B
,9,32):: NEXT B :: RETURN
```

Les nombres en LOGO

Sophie

Nous avons déjà eu amplement l'occasion de manipuler des nombres dans les exemples des articles précédents. Ces nombres servaient à :
- coder (numéro d'appel d'un lutin, numéro de sa forme, code numérique de sa couleur);
- mesurer (des vitesses, par exemple pour les lutins, des longueurs ou des amplitudes de rotations avec la tortue...).

Mais on peut aussi s'en servir tout simplement pour compter.

Un ordinateur, c'est bête, mais pas trop...

Sur le clavier, certaines touches portent les chiffres, et d'autres portent des symboles d'opération ("+", "−", "*" pour la multiplication, et "/" pour la division) et aussi "=", et ".". Attention : prenez l'habitude, pour avoir le plus (+), d'appuyer de la main gauche sur la touche "SHIFT" située à gauche du clavier et de la main droite sur la touche "=". D'abord, c'est une bonne chose que de s'entraîner à utiliser les deux mains, et si possible les dix doigts. Mais il y a plus important : la touche "SHIFT" située à droite du clavier est très proche de la touche "FCTN". Or, FCTN-= vous fait sortir de LOGO, et tout ce que vous aviez emmagasiné dans votre espace de travail est alors perdu, de manière irrémédiable. Quand vous aurez vu partir en un clin d'oeil le travail génial qui vous avait demandé des heures de mise au point, vous comprendrez l'importance de ce conseil d'amie.

Revenons donc à nos moutons, ou plutôt à nos nombres. N'ayez crainte, même si le micro-monde des nombres est un peu plus austère que celui des lutins ou de la tortue, vous ne risquez guère de vous endormir.

C'est au Cours Elémentaire que l'on voit le démarrage de projets sur les nombres. Ayant déjà des habitudes scolaires, les enfants essaient en général d'utiliser la machine comme une calculatrice (ou comme un énoncé d'exercice de calcul), en tapant par exemple :

23+8=

Quelle n'est pas la surprise de voir l'ordinateur faire preuve de mauvaise volonté, et répondre :

PAS ASSEZ DE DONNEES

Certains en déduisent : "c'est comme la chouette (le LITTLE PROFESSOR, ou d'autres calculettes similaires), elle ne fait pas les calculs, mais peut nous dire si on lui donne un résultat juste ou faux". D'autres protestent : "elle n'est pas marrante, ta machine, si c'est nous qui devons faire les calculs !".

Cela engage une série d'activités consistant à préparer des opérations avec leur résultat, et à venir voir l'opinion de la machine. Il se passe encore là quelque chose d'inattendu :

23+8=31

provoque le message :

QUE DOIS-JE FAIRE AVEC VRAI

Remarquons que les élèves ne se découragent pas du tout de recevoir un message :

QUE DOIT-JE FAIRE AVEC FAUX

le cas échéant.

Lorsque les élèves travaillent en groupe, il se met en place très vite une stratégie d'économie que l'on peut exprimer par la règle suivante :

"si j'ai préparé la même opération et trouvé le même résultat que quelqu'un de mon groupe qui est passé à la machine avant moi, je ne réessaie pas" (sous-entendu : on sait bien que la machine ne va pas changer d'avis entre temps).

Par contre, on voit très fréquemment, après qu'un élève ait essayé par exemple :

187+84=231

et obtenu un satisfecit :

QUE DOIS-JE FAIRE AVEC VRAI

plusieurs autres membres du groupe essayer d'autres solutions, telles que :

147+84=221

voire :

147+84=131

et autres, et sembler presque surpris de voir la machine critiquer :

QUE DOIS-JE FAIRE AVEC FAUX

On franchit une étape importante quand l'un des enfants remarque : "Henri a eu VRAI, et moi je n'ai pas le même résultat, alors je n'ai plus de chance, elle va me dire FAUX".

Certains petits malins essaient de se montrer plus astucieux que l'ordinateur, et cherchent le moyen de lui faire faire le calcul. Et contrairement au Basic avec qui "27+18" est effectué, mais reste dans les oubliettes, avec LOGO, on est gratifié d'un rappel à l'ordre ironique :

QUE DOIS-JE FAIRE AVEC 45

Là, c'est la grande stupéfaction : comment, elle est assez maline pour faire le calcul (et même avec les retenues, remarquent certains avec admiration !), et elle demande ce qu'il faut faire avec le résultat ? Vraiment, une grande perte de prestige... pas du tout pour l'ordinateur, qui n'impressionne pas plus la jeune génération qu'un moulin à café électrique, mais plutôt pour les adultes qui font tant d'histoires à propos d'un ordinateur, alors que finalement ce n'est pas grand'chose d'autre qu'un tas de ferraille et de matière plastique, à qui il faut tout dire, même ce qui est évident !

Tout de même, cette réponse incongrue amène certains à réfléchir et à s'interroger : "mais alors, si elle ne sait pas quoi en faire, c'est peut-être qu'elle pourrait en faire autre chose que de l'afficher ?", et à proposer : "elle pourrait peut-être le mettre de côté dans une boîte pour s'en resservir plus tard ?".

Surprise ausi, en primaire, sur les méthodes de calcul de l'ordinateur. En effet, contrairement à l'attente des élèves, "2+3*4" ne donne pas 20 comme résultat (lecture de gauche à droite), mais 14 (utilisation des priorités classiques de l'algèbre). D'où un apprentissage précoce de ces règles par familiarisation grâce à une manipulation volontaire.

Un compte à rebours (où l'on retrouve la récursion)

Nous avions vu, dans l'article précédent, la récursion à propos de la tortue. Nous allons l'utiliser à nouveau pour construire une procédure de compte à rebours.

Que faut-il enseigner à l'ordinateur pour qu'il puisse compter à rebours ? Tout d'abord, il faut afficher la valeur de départ. Autrement dit, notre procédure va ressembler à ceci :

- POUR REBOURS :DEPART
- AFFICHE :DEPART
- FIN

On est évidemment encore loin du compte, car si l'on demande :

- REBOURS 5

on obtient seulement 5.

Autrement dit, il nous faudrait maintenant taper :

- REBOURS 4

c'est-à-dire rappeler la procédure REBOURS avec la valeur :DEPART−1.

Voilà donc l'appel récursif dont nous avons besoin.

Notre procédure devient alors :

- POUR REBOURS :DEPART
- AFFICHE :DEPART
- REBOURS :DEPART−1

Regardons cette nouvelle procédure :
5 4 3 2 1 0 −1 −2 −3 ...

Aïe ! Ca marche un peu trop bien, il nous faut utiliser l'arrêt de panique FCTN-9. En effet, nous n'avons pas indiqué à l'ordinateur à quel moment il doit s'arrêter, et il ne va sûrement pas prendre d'initiative. Or on voudrait que, quand on arrive à zéro, le compte s'arrête. Encore une modification à notre procédure :

- POUR REBOURS :DEPART
- AFFICHE :DEPART
- SI :DEPART=0 ALORS STOP
- REBOURS :DEPART−1
- FIN

Regardons cela :
REBOURS 5
5 4 3 2 1 0
Superbe !

Peut-on faire un compte à rebours de deux en deux ? Peu de changement à apporter à notre précédent raisonnement : il suffit de rappeler la procédure avec :DEPART−2 au lieu de :DEPART−1 ? Faisons une nouvelle procédure :

- POUR REB2 :DEPART
- AFFICHE :DEPART
- SI :DEPART=0 ALORS STOP
- REB2 :DEPART−2
- FIN

Essayons
REB2 10
10 8 6 4 2 0
Trop beau pour être vrai; essayons encore :
REB2 9
9 7 5 3 1 −1 −3 ...

Zut ! Encore un petit pépin, qui vous oblige à avoir recours à nouveau à l'arrêt de panique. Notre test d'arrêt repose sur une égalité, mais celle-ci n'est suceptible de devenir vraie qu'une fois sur deux, en fonction de la valeur de :DEPART. Il nous suffit de la remplacer par une inégalité :

- POUR REB2 :DEPART
- AFFICHE :DEPART
- SI :DEPART<0 ALORS STOP
- REB2 :DEPART−2
- FIN

pour que tout rentre dans l'ordre.

Que se passe-t-il si on modifie l'ordre des lignes d'instructions ?

- POUR REB1 :DEPART
- SI :DEPART=0 ALORS STOP
- AFFICHE :DEPART
- REB1 :DEPART−1
- FIN

(permutation des deux premières lignes de la procédure REBOURS)
REB1 5
5 4 3 2 1

Le zéro n'est pas affiché, puisque le test d'arrêt fait terminer l'exécution avant l'affichage de la valeur zéro.

Et si maintenant on permute les deux dernières lignes ?

- POUR REB :DEPART
- SI :DEPART=0 ALORS STOP
- REB :DEPART−1
- AFFICHE :DEPART
- FIN

on retrouve une procédure avec appel récursif non terminal :
REB 5
1 2 3 4 5
n'a plus rien d'un compte à rebours, puisque l'affichage se fait au moment du dépilage.

Regardons cela de près encore une fois :

REB 5

- La procédure REB s'exécute, :DEPART a la valeur 5. Le test (5=0) a la valeur FAUX, donc cette première ligne est ignorée.
- Il faut alors exécuter REB 4. Cette fois ci, :DEPART vaut 4. Le test (4=0) est encore faux. On trouve donc un nouvel appel.
- REB 3 commence. La valeur de :DEPART est 3. Le test (3=0) est toujours faux. On recommence encore une fois.
- REB 2 s'exécute maintenant, avec la valeur 2 pour :DEPART. Le test (2=0) reste FAUX, donc on a à nouveau un appel.
- REB 1 correspond à la valeur 1 pour :DEPART. Le test (1=0) garde la valeur FAUX, et ceci provoque un nouvel appel à la procédure.
- REB 0 s'exécute alors, avec la valeur 0 pour :DEPART, ce qui donne enfin VRAI pour le test, et amène l'arrêt de cette procédure, qui a, en quelque sorte, terminé son travail et le fait savoir à la procédure qui l'a appelée.
- On se retrouve dans l'exécution de la procédure REB 1, qui s'était arrêtée en cours de route: il reste une ligne à effectuer, l'affichage de la valeur 1, valeur de :DEPART dans cette procédure. La ligne suivante, FIN, indique que cette exécution est terminée.
- Donc REB 2 peut continuer son propre travail, à savoir afficher sa valeur de :DEPART, ici 2. On arrive alors à la FIN de cette procédure.
- REB 3, en affichant sa valeur de :DEPART, soit 3, termine à son tour.
- De même, REB 4 poursuit l'exécution de la ligne d'affichage, et inscrit la valeur 4 qu'a alors :DEPART, dans cette procédure, avant de rendre l'exécution à la procédure appelante.
- REB 5 s'exécute enfin, avec l'affichage de la valeur locale de :DEPART, à savoir 5. Cette procédure terminée, on se trouve ramené au niveau supérieur.

Sortez-les !

Regardons maintenant quelque chose de légèrement différent, avec des procédures pour calculer le CARRE et le CUBE d'un nombre.

Cela semble très simple. Bien sûr, le carré d'un nombre est le produit de ce nombre par lui-même, et le cube d'un nombre est le produit du carré par le nombre (ce qui nous offrira l'occasion d'utiliser une procédure comme sous-procédure d'une autre procédure).

- POUR CARRE :NOMBRE
- AFFICHE :NOMBRE⋆ :NOMBRE
- FIN

semble fonctionner parfaitement :
CARRE 4
16
CARRE 120
14400
CARRE −5
25

Continuons :

- POUR CUBE :VALEUR
- AFFICHE
  CARRE :VALEUR⋆ :VALEUR
- FIN

Là, nous commençons à avoir des ennuis.
Tout d'abord le calcul ne se fait pas comme escompté. Si on demande CUBE 3, le nombre mentionné est 81, qui n'est pas le cube de 3, mais sa puissance quatrième.

Cela provient d'une règle :

Toute opération infixée est effectuée avant les opérations préfixées.

Pour ceux que les qualificatifs infixé et préfixé laissent perplexes, comparons deux façons d'indiquer une addition :

2+3
ou
SOMME 2 3

Dans le premier cas, l'opération est indiquée par un symbole écrit entre les deux nombres sur lesquels porte l'opération. On dit qu'il s'agit d'une notation infixée. Dans le deuxième cas, l'opération est indiquée par un mot qui précède les nombres sur lesquels porte l'opération. On dit qu'il s'agit d'une notation préfixée.

Par conséquent, dans :

- CARRE :VALEUR⋆ :VALEUR

la multiplication (infixée) est effectuée avant l'élévation au carré (préfixée), ce qui explique que l'on trouve la puissance quatrième du nombre fourni.

Deux manières de remédier à cela :

- ou bien parenthéser :
- (CARRE :VALEUR)⋆ :VALEUR
- ou bien inverser l'ordre :
- :VALEUR⋆CARRE :VALEUR

Cette modification ne nous sort pas

complètement de notre problème, car maintenant le système se plaint que le CARRE N'A RIEN SORTI.

Regardons de plus près ce que nous avons mis dans notre procédure CARRE :

- POUR CARRE :NOMBRE
- AFFICHE :NOMBRE* :NOMBRE
- FIN

Non seulement nous avons demandé à la procédure d'effectuer le calcul, mais en plus nous lui avons indiqué qu'elle doit afficher le résultat sur l'écran. Et comme l'ordinateur est bête et discipliné, il nous obéit aveuglément.
Maintenant, essayez de vous imaginer en train de faire de l'affichage (selon vos habitudes, vous vous référerez à du collage d'affiches sur des panneaux électoraux ou à du punaisage de posters sur les murs de votre chambre !). Au début de l'exercice, vous avez en main l'affiche et de quoi la coller (une brosse et un pot de colle, ou des punaises, ou du ruban adhésif et une paire de ciseaux, suivant les circonstances). Quand l'opération est terminée, l'affiche n'est plus entre vos mains, mais sur le mur, et il ne vous reste plus dans les doigts que le nécessaire de fixation.

Hé bien ! C'est exactement ce qui se passe ici. CARRE ayant affiché son résultat, il n'y a plus rien qui puisse être transmis pour le calcul suivant. Et le message, en même temps qu'il vous signale cette anomalie, vous souffle aussi comment vous en sortir ! Il suffira de modifier nos deux procédures :

- POUR CARRE :NOMBRE
- SORS :NOMBRE* : NOMBRE
- FIN

et

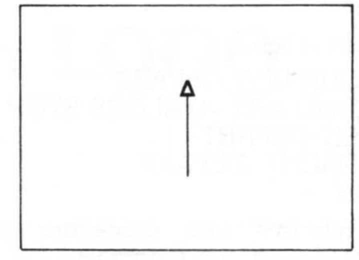
- POUR CUBE :VALEUR
- SORS :VALEUR*CARRE : VALEUR
- FIN

Bien entendu, maintenant, exactement comme lorsqu'on demandait directement un calcul, il va falloir penser à préciser ce qu'on souhaite que l'ordinateur fasse avec le résultat, sinon :
CUBE 4
QUE DOIS-JE FAIRE AVEC 64

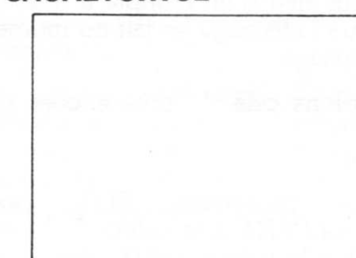
Même si ça vous épate, la tortue sait compter.
On peut, comme nous venons de le faire, jouer avec les nombres pour le plaisir. Mais on peut aussi voir comment la tortue s'en débrouille.

On regarde d'abord l'addition. Deux manières principales de se convaincre que notre petite bête sait ajouter :

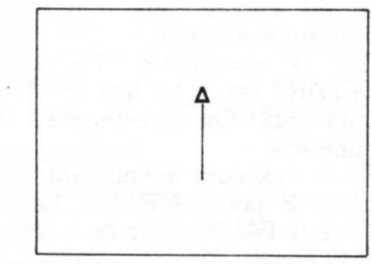
- EFFACEECRAN
- AVANCE 30+25



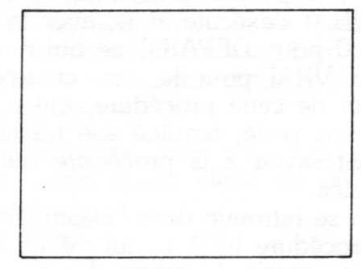
- EFFACEUR
- RECULE 55
- CACHETORTUE



- POSEPLUME
- MONTRETORTUE
- AVANCE 30+25

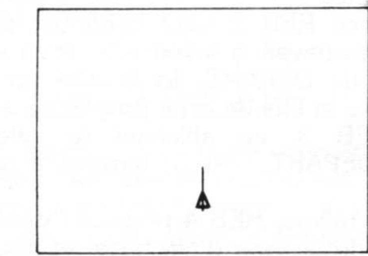


- EFACEUR
- RECULE 30
- RECULE 25
- CACHETORTUE



Voyons maintenant avec les soustractions. Au début, cela semble parfait :

- POSEPLUME
- MONTRETORTUE
- AVANCE 40−25



- EFFACEUR
- RECULE 15
- CACHETORTUE

Mais il y a des moments d'inquiétude, quand on essaie :
AVANCE 30−55
car là, oh stupeur ! elle recule.

Première réaction : "la machine est en panne". On recommence, et on s'aperçoit qu'il s'agit là d'un comportement persistant. Même chose pour DROITE 30−120, elle pivote vers la GAUCHE. Il faut donc, pour les élèves de l'école primaire (qui ignorent en général tout des nombres relatifs - les nombres en-dessous de zéro des températures n'ayant pas du tout le même statut que les nombres scolaires -) qui se sont aventurés sur ce terrain glissant, élargir le savoir antérieur pour y incorporer cette acquisition.

Certains en profitent pour poursuivre une exploration sur les nombres :
30−55
ou 20−140
et découvrir ainsi les "nombres à tiret" ou "nombres à moins", et les règles d'opérations qui les concernent.

Bien que la version TI-LOGO ne comporte que les entiers relatifs (de −32767 à 32768), il y a déjà là un vaste micro-monde à explorer. En fait, il est possible d'utiliser dans des calculs des valeurs relationnelles et même des racines carrées, mais cette astuce constitue une autre histoire, que nous vous raconterons une autre fois. Avec les thèmes que nous venons d'aborder ensemble, et avec la fonction aléatoire, on peut déjà s'offrir de quoi meubler agréablement de longues soirées en toute saison.

Sur ce thème des nombres, vous trouverez un chapitre détaillé dans :

Une tortue dans une classe - de Catherine Berdonneau et Rose-Marie Dumas - cahier PACIFIC No 1 - 1982 (le matériel utilisé est un micro-ordinateur qui n'est plus fabriqué depuis plusieurs années, mais les observations restent toujours valables).

Signalons également un compte-rendu d'activités réalisées sur le TI-99/4A avec des enfants en situation d'échec scolaire :

Une semaine d'activités LOGO en C.P.P.N. - de Catherine Berdonneau et Pierre Boyer - Document de travail No 7 - GREPACIFIC - 1983.

Ces deux fascicules sont disponibles auprès de GREPACIFIC - 51 Boulevard des Batignolles - 75008 PARIS (adresse postale uniquement). Le cahier No 1 coûte 60 F, le document de travail No 7 coûte 30 F. Seules les commandes accompagnées d'un chèque à l'ordre de GREPACIFIC en règlement peuvent être traitées.
A bientôt... ■

Graphisme en haute résolution sur le TI-99

Gérard Santraille et Jean-Luc Bazanegue

Notre TI-99 étant dépourvu d'instructions permettant de manipuler simplement, à partir du Basic TI, des graphiques en haute définition (résolution de 256 colonnes et 192 lignes soit 49152 pixels), nous vous présentons un programme en assembleur constitué par un ensemble de routines utilisables séparément à partir du BASIC-TI (avec le module Minimem ou l'Editeur/Assembleur et l'extension de mémoire 32K).

En effet, le simple tracé d'une droite en haute résolution à partir du Basic seul n'est réalisable qu'au moyen de l'instruction CHAR, et seulement après plusieurs pages de calculs. En Basic, pour positionner un point en haute résolution, il faut :

1) déterminer la position du caractère basse résolution (ligne et colonne) où il se positionne;
2) connaître le descripteur du caractère en question (ce qui suppose une gestion complète de tous les descripteurs si l'on travaille en Basic TI, qui ne supporte pas l'instruction "CHARPAT");
3) modifier le descripteur du caractère en "noircissant", dans la matrice 8x8 qui le compose, le pixel (point élémentaire) adéquat.

Cette méthode, on s'en rend aisément compte, est très lourde; elle est de plus extrêmement lente et ne permet d'afficher qu'un nombre limité de points. En effet, le Basic ne permet de redéfinir que 128 caractères; d'autre part, la couleur de chacun de ces caractères n'est pas indépendante.

Plusieurs d'entre vous nous ont proposé un programme Basic permettant, par la méthode que je viens d'exposer, de tracer une courbe en haute résolution. Ces programmes mettent trente minutes pour tracer un simple sinus... et ne peuvent superposer une parabole rouge à un cercle jaune et une ellipse verte... (on pourrait ainsi varier indéfiniment les formes et les couleurs).

Nous vous proposons cependant un petit programme, exécutable en Basic TI avec la console seule, qui peut servir de base à la création d'un programme de tracé plus élaboré. Ce programme permet de sauvegarder sur disquette la courbe obtenue.

```
100 REM ******************
110 REM *   Tracage  de   *
120 REM *   courbes  en   *
130 REM *haute resolution *
140 REM ******************
150 REM *    Basic  TI    *
160 REM ******************
170 REM *    Copyright    *
180 REM *  "99 Magazine"  *
190 REM *Gerard Santraille*
200 REM *   et Jean-luc   *
210 REM *    Bazanegue    *
220 REM ******************
230 REM
240 CALL SCREEN(2)
250 CALL CLEAR
260 PRINT "*****************************
  tracage de courbe     **   en haute
resolution   *"
270 PRINT "*definition: 192x256 pixels***
*************************": : : :,"CHO
ISISSEZ"
280 PRINT : :" T)RACER UNE FONCTION": :"
L)IRE UN TRACE": : :
290 CALL SCREEN(15)
300 CALL SOUND(100,700,0)
310 CALL KEY(5,H,S)
320 IF (H-84)*(H-76)THEN 310
330 CALL CLEAR
340 IF H-76 THEN 630
350 PRINT "  Lecture d'un trace.....": :
:
360 INPUT "Nom du fichier : ":F$
370 CALL CLEAR
380 CALL SCREEN(13)
390 GOSUB 1630
400 OPEN #1:F$,INTERNAL,FIXED 19
410 INPUT #1:C$
420 CALL CHAR(33,C$)
430 INPUT #1:C$
440 CALL CHAR(34,C$)
450 INPUT #1:C$
460 CALL CHAR(35,C$)
470 L1=ASC(SEG$(C$,17,1))
480 C1=ASC(SEG$(C$,18,1))
490 CALL HCHAR(L1,1,33,32)
500 CALL VCHAR(1,C1,34,24)
510 CALL HCHAR(L1,C1,35)
520 K=36
530 INPUT #1:C$
540 CALL CHAR(K,C$)
550 L1=ASC(SEG$(C$,17,1))
560 C1=ASC(SEG$(C$,18,1))
570 CALL HCHAR(L1,C1,K)
580 K=K+1
590 IF EOF(1)=0 THEN 530
600 CLOSE #1
610 GOSUB 1670
620 GOTO 620
630 PRINT "  Trace d'une fonction": : :,"
Patientez...."
640 DEF F(X)=EXP(-X/5)*SIN(X)
650 DIM PAT$(127),T(3)
660 FOR I=0 TO 127
670 PAT$(I)="0000000000000000"
680 NEXT I
```

```
690 REF$="0123456789ABCDEF"
700 CALL HCHAR(23,17,32,13)
710 INPUT "Pas du trace : ":P
720 IF (P<.05)+(P>20)THEN 710
730 PRINT : :
740 INPUT "Xmin,Xmax,Xaxe: ":XI,XA,XX
750 IF (XI<XA)*(XX<=XA)*(XX>=XI)THEN 790
760 CALL SOUND(200,200,0)
770 PRINT ,"INCORRECT"
780 GOTO 740
790 INPUT "Fmin,Fmax,Faxe: ":YI,YA,YX
800 IF (YI<YA)*(YX<=YA)*(YX>=YI)THEN 840
810 CALL SOUND(200,200,0)
820 PRINT ,"INCORRECT"
830 GOTO 790
840 CALL CLEAR
850 CALL SCREEN(13)
860 GOSUB 1630
870 DEF HL(X)=INT(1+191*(YA-X)/(YA-YI))
880 DEF HC(X)=INT(1+255*(X-XI)/(XA-XI))
890 DEF Q1(X)=INT((X-1)/8)+1
900 DEF Q2(X)=X-8*(Q1(X)-1)
910 L=HL(YX)
920 C=HC(XX)
930 L1=Q1(L)
940 L2=Q2(L)
950 C1=Q1(C)
960 C2=Q2(C)
970 PAT$(1)=SEG$(PAT$(1),1,2*L2-2)&"FF"&S
EG$(PAT$(1),2*L2+1,14)
980 IF C2>4 THEN 1010
990 R$=STR$(2^(4-C2))&"0"
1000 GOTO 1020
1010 R$="0"&STR$(2^(8-C2))
1020 PAT$(2)=R$&R$&R$&R$&R$&R$&R$&R$
1030 PAT$(3)=SEG$(PAT$(2),1,2*L2-2)&"FF"&
SEG$(PAT$(2),2*L2+1,14)&CHR$(L1)&CHR$(C
1)
1040 CALL CHAR(33,PAT$(1))
1050 CALL CHAR(34,PAT$(2))
1060 CALL CHAR(35,PAT$(3))
1070 CALL HCHAR(L1,1,33,32)
1080 CALL VCHAR(1,C1,34,24)
1090 CALL HCHAR(L1,C1,35)
1100 DD=(XA-XI)*P/256
1110 K=36
1120 U=XI
1130 FOR X=1 TO 256 STEP P
1140 Y=HL(F(U))
1150 U=U+DD
1160 X1=Q1(X)
1170 X2=8*X1-INT(X)
1180 Y1=Q1(Y)
1190 Y2=2*Q2(Y)
1200 IF (Y1<1)+(Y1>24)THEN 1440
1210 CALL GCHAR(Y1,X1,COD)
1220 IF COD>34 THEN 1270
1230 PAT$(K-32)=PAT$(COD-32)
1240 COD=K
1250 IF K=160 THEN 1450
1260 K=K+1
1270 CO$=SEG$(PAT$(COD-32),1,16)
1280 IF X2<4 THEN 1310
1290 X2=X2-4
1300 Y2=Y2-1
1310 S$=SEG$(CO$,Y2,1)
1320 S=POS(REF$,S$,1)-1
1330 T(3)=INT(S/8)
1340 T(2)=INT((S-8*T(3))/4)
1350 T(1)=INT((S-8*T(3)-4*T(2))/2)
1360 T(0)=S-8*T(3)-4*T(2)-2*T(1)
1370 IF T(X2)THEN 1400
1380 S=S+2^X2
1390 S$=SEG$(REF$,S+1,1)
1400 CO$=SEG$(CO$,1,Y2-1)&S$&SEG$(CO$,Y2+
1,16-Y2)
1410 CALL CHAR(COD,CO$)
1420 PAT$(COD-32)=CO$&CHR$(Y1)&CHR$(X1)
1430 CALL HCHAR(Y1,X1,COD)
1440 NEXT X
1450 GOSUB 1670
1460 CALL KEY(3,H,S)
1470 IF H-12 THEN 1460
1480 CALL CLEAR
1490 PRINT "Tapez 'CON' pour poursuivre":
:
1500 BREAK
1510 CALL CLEAR
1520 CALL SCREEN(15)
1530 PRINT "  Sauvegarde du trace...": :
:
1540 INPUT "Nom du fichier : ":F$
1550 OPEN #1:F$,INTERNAL,FIXED 19
1560 FOR I=33 TO K-1
1570 PRINT #1:PAT$(I-32)
1580 NEXT I
1590 CLOSE #1
1600 GOSUB 1670
1610 CALL CLEAR
1620 END
1630 FOR I=1 TO 16
1640 CALL COLOR(I,12,1)
1650 NEXT I
1660 RETURN
1670 FOR I=1 TO 4 STEP .5
1680 CALL SOUND(40,300*I,0)
1690 NEXT I
1700 RETURN
```

Pour s'affranchir de toutes ces limitations, entrons dans le mode "BIT-MAP". Accrochez vos ceintures, le voyage commence...

Sans donner trop de détails sur le fonctionnement propre du programme que nous vous proposons (voir explications pages suivantes et sur le listing du fichier-source), il convient d'éclaircir certains points sur la structure et l'organisation interne de la mémoire du TI-99. Notre ordinateur utilise deux zones de mémoire vive différentes :

1) La mémoire CPU (Central Processing Unit) directement adressable par le microprocesseur. Cette

mémoire se situe physiquement dans la carte d'extension, mis à part une zone de 256 octets située à l'intérieur du microprocesseur et qui constitue, en configuration de base, la seule mémoire directement adressable par le TMS 9900.

2) La mémoire VDP (Video Display Processing), qui est utilisée par le processeur vidéo pour la gestion de l'écran. Il s'agit là de la mémoire résidente de la console.

La mémoire VDP forme un ensemble de 16K contenant notamment la table écran (un tableau de 768 octets indiquant le code ASCII de chaque caractère affiché), la table des couleurs (16 groupes ayant chacun 2 couleurs soit 16 octets), la table de définitions de tous les caractères (512 octets)... etc... le programme Basic. Pour avoir la partition exacte de la mémoire VDP lors d'une utilisation avec l'interpréteur Basic, se référer au manuel de la Mini-mémoire ou à celui de l'Editeur/Assembleur.
La mémoire VDP n'est pas adressée directement par le microprocesseur, ceci explique en partie la relative lenteur du Basic.
Le mode Bit-Map utilise une grande partie de la mémoire vidéo (13056 octets); il n'est donc pas possible de faire cohabiter ce mode d'affichage et le Basic. Donc, lors de la conception du programme, nous étions confrontés au problème suivant : accéder à la haute résolution tout en conservant la souplesse et la facilité d'emploi du Basic.

Nous avons contourné le problème de la façon suivante :

Le graphique est construit point par point dans deux tampons de la mémoire CPU (6K pour la création de l'image et 6K pour la sauvegarde des couleurs). Jusqu'à la visualisation de l'image, le programme Basic reste donc toujours localisé dans la mémoire VDP (qui est, répétons-le, son emplacement normal).
Lors de l'affichage du graphique par une routine (nous verrons plus loin comment l'utiliser), toute la mémoire VDP est sauvegardée dans un autre tampon de 16K en mémoire CPU, et les deux tampons de 6K dont nous avons parlé plus haut sont transférés en mémoire VDP.
L'image apparaît, et le programme Basic n'est pas perdu puisqu'il a été sauvegardé en mémoire CPU.
Lorsque l'image est visualisée, on peut revenir au programme Basic en pressant "BACK" (FCTN-9). Le tampon de 16K réintègre sa position et le programme Basic reprend son cours.
En jonglant ainsi avec les tampons, entre la mémoire CPU et la mémoire VDP, on accède très simplement à la haute résolution.

Voyons maintenant comment utiliser les différentes routines graphiques, et certaines applications des nouvelles capacités de notre TI-99.
Avant toute utilisation de l'une des routines, il convient de charger le programme en Assembleur en mémoire CPU. Si vous possédez le module Editeur/Assembleur, donc un lecteur de disquettes, votre programme doit débuter par :

CALL LOAD("DSK1.GRAPH/OBJ")

à condition, bien sûr, d'avoir choisi "GRAPH/OBJ" comme nom de code objet au moment de l'assemblage.

Note : il n'est pas nécessaire de placer cette instruction dans le programme, on peut très bien l'exécuter en mode commande. Les commandes NEW et BYE ne l'effacent pas puisqu'il est implanté en mémoire CPU.

Si vous n'avez pas de lecteur de disquettes et que vous voulez exploiter cet ensemble de routines à partir de la Mini-mémoire, il convient de charger le programme Basic baptisé "Code", qui "poke" directement le code objet en mémoire. Comme pour le cas précédent, une fois que le programme graphique a été chargé, NEW ou BYE ne l'affecte pas.
La lecture du fichier source (dont le listing est donné plus loin) montre qu'il existe six références externes (étiquettes à partir desquelles on peut lancer l'exécution depuis le Basic). Ces références correspondent à des points d'entrée dans le programme en Assembleur. Dans un but de clarté, nous ne parlerons plus de points d'entrée, mais de routines. Nous allons voir leurs rôles et syntaxes.

EFFT, EFFP, EFFC, AXES, POINT et IMAGE

Routine "EFFT"

Cette routine, qui signifie EFFace Tout, nettoie complètement l'écran graphique. Sa syntaxe est :

CALL LINK("EFFT")

Si vous n'employez pas cette instruction lors de la création d'une nouvelle image, cette dernière se superposera à la précédente. On peut ainsi créer, morceau par morceau, des images graphiques complexes.

Routines "EFFP" et "EFFC"

L'action de "EFFP" (pour EFFace la table des Points) et de "EFFC" (pour EFFace la table des Couleurs) est plus sélective que celle de EFFT :

CALL LINK("EFFP")

nettoie la table de définition de l'image en plaçant la valeur zéro (0) dans tous les octets du tampon de 6K qui définit la position de chaque point.

CALL LINK("EFFC")

a la même action que "EFFP", mais son effet porte sur l'autre tampon de 6K, celui qui définit la couleur des points affichés.

Routine "AXES"

Lors du tracé de fonctions (qu'elles soient cartésiennes, paramétriques ou polaires), il est toujours intéressant de pouvoir placer des axes sur l'écran graphique. Il aurait été possible de dessiner des axes point par point au moyen de la routine "POINT", décrite un peu plus loin, mais nous avons préféré écrire une petite routine en Assembleur qui fait cela automatiquement, permettant ainsi à l'utilisateur de se concentrer uniquement sur la fonction à dessiner.
Nous avons toujours évité de passer des arguments du Basic à l'Assembleur au moyen de LINK (économie de temps et emploi plus souple). Ainsi, quand une routine nécessite des arguments (c'est le cas de "AXES", de "POINT" et de "IMAGE"), il faut les "poker" directement du Basic (par le sous-programme "LOAD") avant d'appeler la routine par "CALL LINK" :

CALL LOAD(−24570,XPOS,YPOS)
CALL LINK("AXES")

Dans le cas présent, la routine "AXES" doit lire deux valeurs, la position de chaque axe. Il convient donc de faire CALL LOAD(−24570,XPOS,YPOS), où XPOS est une valeur comprise entre 0 et 191 (axe des abscisses, horizontal) et YPOS une valeur comprise entre 0 et 255 (axe des ordonnées, vertical).
Ainsi, pour tracer les axes au beau milieu de l'écran, on peut écrire :

CALL LOAD(−24570,95,128)
CALL LINK("AXES")

Petites précisions au sujet des passages de paramètres par "CALL LOAD" :

L'instruction (ou commande) "CALL LOAD" accepte, comme valeur à "poker", tous les nombres compris entre −32768 et 32767, même si ces nombres ne sont pas des entiers. Puisque le paramètre passé doit "tenir" dans un octet (8 bits), la valeur résultant de l'opération sera toujours un entier compris entre 0 à 255.

Exemple :
CALL LOAD(−20000,500.49)
La valeur 500.49 est d'abord réduite à l'entier le plus proche, il nous reste donc 500.

En binaire, et en complément à deux sur un mot mémoire (16 bits soit 2 octets), 500 est représenté de la manière suivante :
500 = 00000001 11110100 (>01F4 en hexa)
A ce stade, l'interpréteur Basic supprime l'octet de poids fort pour ne conserver que l'octet de poids faible. La valeur "pokée" sera donc : 11110100 en binaire, >F4 en hexadécimal, et 244 en décimal.
Si vous désirez obtenir des détails supplémentaires au sujet des mots, octets et des différents codages des nombres, vous pouvez vous reporter à l'article "Les modes d'adressage en Assembleur", paru dans le numéro 3 de "99 Magazine".
Që TI4 2
Pour l'axe horizontal, la routine contrôle la validité de la valeur. Si la position est supérieure à 191, la valeur prise par défaut sera 191 (première ligne de l'écran, en partant du haut).
On peut choisir la couleur des axes tracés en "pokant" une valeur aux adresses −24574 et −24573. En effet, cette adresse (qui, comme nous le verrons plus loin est également utilisée par la routine "POINT") contient la couleur des points "OFF" suivie de celle des points "ON".
Rappelons que le mode Bit-Map ne permet pas de définir de façon totalement indépendante la couleur de chaque pixel. L'écran est découpé en petits tronçons horizontaux de 8 pixels, dont on peut définir la couleur des points "ON" (allumés) et celle des points "OFF" (éteints). La couleur "OFF" se situe à l'adresse −24574 et la couleur "ON" à l'adresse −24573.

ATTENTION : le numéro de la couleur varie entre 0 et 15 et non pas entre 1 et 16 comme en Basic. Ainsi, CALL LOAD(−24574,0,13) indique que le tracé doit se faire en Magenta sur un fond transparent (donc de la couleur de l'écran).

Routine "POINT"

C'est la routine fondamentale de cet ensemble. Elle positionne le point dont les coordonnées sont "pokées" en −24576 pour l'abscisse (comprise entre 0 et 255) et −24575 pour l'ordonnée (comprise entre 0 et 191). Si l'abscisse est supérieure à 191, la routine redonne le contrôle au Basic en ignorant le point.

Comme la routine "AXES", "POINT" va lire aux adresses −24574 et −24573, respectivement, les couleurs "OFF" et "ON" du tronçon de 8 pixels dans lequel se trouve le point en question. Avec cette possibilité, il est possible de superposer plusieurs courbes de couleurs différentes.

La syntaxe est la suivante :

CALL LOAD(−24574,COFF,CON)
pour le choix des couleurs

CALL LOAD(−24576,X,Y)
pour la position du point

CALL LINK("POINT")
pour l'appel de la routine

Note : il est possible de rassembler les deux CALL LOAD :

CALL LOAD(−24574,X,Y,COFF,CON)

Une fois définies, les couleurs "ON" et "OFF" sont prises par défaut, sauf si l'on "poke" de nouvelles valeurs aux adresses −24574 et −24573. Ainsi :

```
100 CALL LOAD(-24574,0,6)
110 FOR X=0 TO 255
120 CALL LOAD(-24576,X,X/2)
130 NEXT X
```

trace une droite rouge foncé sur fond transparent, donc de la couleur de l'écran. Il est ainsi possible de garder une ou plusieurs valeurs par défaut. Rappelons que ceci n'est possible que parce que les arguments ne sont pas "passés" lors du LINK, mais sont directement "pokés" au moyen de CALL LOAD.

Routine "IMAGE"

Lorsque la construction de l'image est terminée, il convient de l'afficher en la transférant dans la mémoire VDP. Cette routine effectue deux tâches principales :

1) Sauvegarde de toute la mémoire VDP (donc le programme Basic) dans un tampon de 16K de la mémoire CPU.
2) Transfert de l'image (deux tampons de 6K) en mémoire vidéo et visualisation quasi instantanée.

C'est avant d'appeler cette routine que l'on peut choisir la couleur de l'écran qui doit être "pokée" en −24572 (rappelez-vous que la valeur de la couleur doit être comprise entre 0 et 15 et non pas entre 1 et 16).

```
CALL LOAD(-24574,15)
CALL LINK("IMAGE")
```

affiche l'image sur un écran blanc.

Lorsque l'image est visualisée, il est possible de revenir au Basic en pressant "BACK" (FCTN-9).

Voilà... vous savez tout. Votre TI-99 sait désormais tracer en haute résolution.

Quelques remarques et conseils d'utilisation.

L'écran de votre TI-99 n'est pas "orthonormé" : en effet, la hauteur d'un pixel est environ 1,35 fois plus petite que sa largeur. Tenez-en compte lors de vos tracés afin que vos cercles ne ressemblent pas trop à des ellipses.
Pour tracer une série de cercles concentriques (de couleurs différentes) dignes de ce nom, on peut écrire :

```
100 REM ***************
110 REM *   Cercles   *
120 REM *concentriques *
130 REM ***************
140 REM *  Execution:  *
150 REM *    16mn 20s  *
160 REM ***************
170 REM
180 CALL LINK("EFFT")
190 DATA 1,4,6,11,12,13
200 CALL LOAD(-24574,0,1,15)
210 FOR R=10 TO 60 STEP 10
220 READ C
230 CALL LOAD(-24573,C)
240 P=.5/R
250 RY=1.35*R
260 FOR T=0 TO 6.2832 STEP P
270 CALL LOAD(-24576,128+R*COS(T
),95+RY*SIN(T))
280 CALL LINK("POINT")
290 NEXT T
300 NEXT R
310 CALL LINK("IMAGE")
```

N'oubliez pas de charger le programme en assembleur avant de vous servir d'une de ses routines... (ne riez pas, cela nous est souvent arrivé !).
De par la limitation en blocs horizontaux de huit pixels imposée par le découpage, il peut arriver que lorsque deux courbes de couleurs différentes se croisent (ou lorsque une courbe croise un axe), leurs couleurs se confondent (la couleur de la dernière tracée l'emporte).
La relative lenteur de certains tracés (relative si l'on compare au temps nécessaire pour obtenir en Basic des figures à peu près semblables, vous pourrez d'ailleurs faire la comparaison avec le programme Basic que nous vous proposons.) est imputable aux sous-programmes "LINK" et "LOAD". La routine "POINT" est, quant à elle, quasi instantanée.
Pour tracer une fonction exprimée en coordonnées cartésiennes, c'est-à-dire dont l'équation est du type Y=F(X), ou une courbe exprimée en coordonnées paramétriques (X=F(t) et Y=G(t)), il est plus rapide de faire les calculs directement dans le "CALL LOAD" qui "poke" les valeurs de l'abscisse et de l'ordonnée.
Le programme suivant trace une rosace selon cette méthode :

```
100 REM ***************
110 REM *    Rosace    *
120 REM ***************
130 REM *  Execution:  *
```

```
140 REM *   13mn 32s   *
150 REM ****************
160 REM
170 CALL LINK("EFFT")
180 CALL LOAD(-24574,0,11,12)
190 FOR T=0 TO 3.1516 STEP .0025
200 CALL LOAD(-24576,128+90*COS(
T)*SIN(5*T),95+60*SIN(T)*SIN(5*T
))
210 CALL LINK("POINT")
220 NEXT T
230 CALL LINK("IMAGE")
```

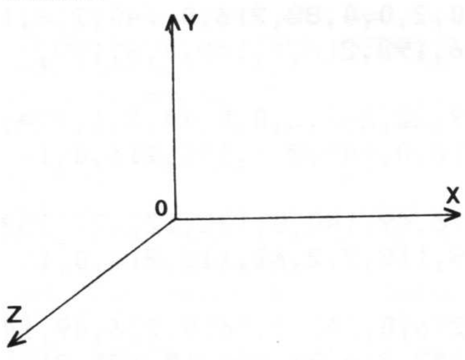
La rosace est exprimée en coordonnées cartésiennes, pour t variant entre 0 et PI, selon :

X=a.Cos(t).Sin(nt)
Y=b.Sin(t).Sin(nt)

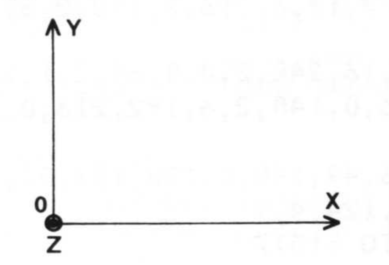
où a et b représentent respectivement la largeur et la hauteur de la rosace et n le nombre de branches.

Note : l'origine de l'écran (0,0) étant située en bas et à gauche (origine classique d'un graphe), lorsque l'on veut centrer une figure géométrique exprimée en coordonnées paramétriques, il convient de rajouter 128 à X et 95 à Y (changement d'origine au centre de l'écran).

Avec cet ensemble de routines, il est possible de simuler la troisième dimension en donnant une impression de profondeur (relief). Pour ce faire, considérons le référentiel direct (qui répond à la règle des trois doigts) suivant :



Cette représentation plane du trièdre "triche": on regarde ce repère sous un certain angle. En effet, dans une représentation rigoureuse, l'axe Z sort de la feuille "en allant vers nous" comme le montre la figure suivante :



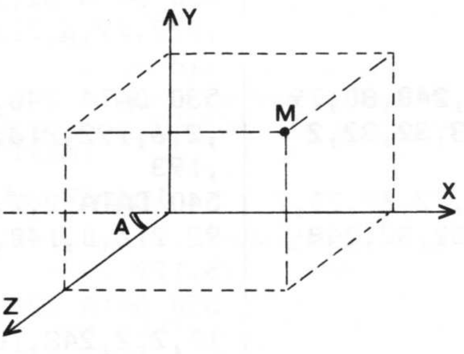
Soit une courbe (C), paramétrée selon :

X=f(t)
Y=g(t)
Z=h(t)

L'écran de notre TI ne peut appréhender que X et Y. Pour reporter la cote Z, il convient de passer du référentiel de la figure 2 à celui de la figure 1. Intuitivement, on "voit" bien que plus le dessin sera proche de nous (c'est-à-dire plus Z sera grand), plus il glissera vers le bord inférieur gauche.
Sur le schéma suivant, on voit qu'un point M de coordonnées (x,y,z) sera reporté à la position :

x−z.Cos(A)
y−z.Sin(A)

Nous tenons ici un procédé très simple pour dessiner une courbe en trois dimensions : sa cote intervient dans l'abscisse et l'ordonnée.
Plus l'angle A est important, plus on a l'impression de regarder la courbe "de haut" (ce qui se traduit mathématiquement par un décalage de plus en plus important sur l'axe des ordonnées, puisque le sinus est une fonction croissante jusqu'à PI/2).
Prenons un exemple extrêmement classique : la courbe de LISSAJOU.
Selon le référentiel de la figure 3, l'équation d'une telle courbe est :



x=R.Cos(t)
y=H.Sin(n.t)
z=R.Sin(t)

où R est le rayon de la courbe, H la hauteur et n le nombre de branches.

Supposons maintenant que l'angle sous lequel on regarde le repère soit de 45 degrés (PI/4). Sachant que Cos(45)=Sin(45)=0.707, on traduit la courbe sur une surface plane selon :

x=R⋆(Cos(t)−0.707⋆Sin(t))
y=H⋆Sin(t)−0.707⋆R⋆Cos(t)

Pour que la courbe utilise une grande partie de l'écran, on prendra R=50 (l'unité est le pixel), H=30 et 10 branches. Comme nous l'avons déjà vu, l'écran étant un peu aplati, nous multiplierons y par 1,35. Enfin, pour centrer l'origine du dessin au milieu de l'écran, nous ajouterons 128 à X et 95 à Y. Finalement, l'équation directement exploitable par notre TI-99 est devenue :

x=128+50⋆(Cos(t)−0.707⋆Sin(t))
y=95+1.35⋆30⋆Sin(10t)
−1.35⋆50⋆0.707⋆Cos(t)

soit :

x=128+50⋆Cos(t)−35.35⋆Sin(t)
y=95+40.5⋆Sin(10t) −47.72⋆Cos(t)

Le programme suivant dessine ce motif en vert foncé sur un écran jaune clair (il s'agit de deux couleurs qui s'assemblent d'ailleurs fort bien).

```
100 REM ***************
110 REM *   Lissajou   *
120 REM ***************
130 REM
140 CALL LOAD(-24574,0,12,11)
150 FOR T=0 TO 6.2832 STEP .0025
160 CALL LOAD(-24576,128+50*COS(
T)-35.35*SIN(T),95+40.5*SIN(10*T
)-47.72*COS(T))
170 CALL LINK("POINT")
180 NEXT T
190 CALL SOUND(500,700,0)
200 CALL LINK("IMAGE")
```

Nous vous proposons un dernier programme, qui trace deux ellipses multicolores. Chaque point de ces ellipses est tranformé en une ligne horizontale de trente points. Vous devrez être patient, car il faut un peu plus d'une heure pour obtenir l'image, mais le résultat est intéressant.

```
100 REM ****************
110 REM *   Ellipses    *
120 REM * multicolores *
130 REM ****************
140 REM
150 CALL INIT
160 CALL LOAD(-24572,1)
170 CALL LOAD("DSK1.OBJET")
180 CALL LINK("EFFT")
190 BE1=3.1416/180
200 BE2=BE1*362
210 BE3=BE1/2
220 FOR BE=BE1 TO BE2 STEP BE3
230 X=111+100*COS(BE)
240 Y=90+90*SIN(BE)
250 RANDOMIZE
260 IF X)111 THEN 290
270 CALL LOAD(-24574,0,INT(RND*7
)+2)
280 GOTO 300
290 CALL LOAD(-24574,0,INT(RND*7
)+9)
300 B1=X
310 B2=X+30
320 FOR B=B1 TO B2
330 CALL LOAD(-24576,B,Y)
340 CALL LINK("POINT")
350 NEXT B
360 NEXT BE
```

```
370 FOR BE=BE1 TO BE2 STEP BE3
380 X=111+60*COS(BE)
390 Y=90+60*SIN(BE)
400 RANDOMIZE
410 IF X<=111 THEN 440
420 CALL LOAD(-24574,0,INT(RND*7
)+2)
430 GOTO 450
440 CALL LOAD(-24574,0,INT(RND*7
)+9)
450 B1=X
460 B2=X+30
470 FOR B=B1 TO B2
480 CALL LOAD(-24576,B,Y)
490 CALL LINK("POINT")
500 NEXT B
510 NEXT BE
520 CALL LINK("IMAGE")
```

Vous disposez, avec cet ensemble de routines, d'un outil puissant et surtout très facile à mettre en oeuvre. Nous parlerons, dans de prochains articles, d'autres applications envisageables. En attendant, n'ayez plus aucune retenue graphique et adressez-nous vos plus belles réalisations. ■

```
100 REM ******************
110 REM *   Ce programme  *
120 REM * implante le code *
130 REM * objet en memoire *
140 REM ******************
150 REM * Basic TI + Mini- *
160 REM *     memoire +    *
170 REM *  extension 32K   *
180 REM ******************
190 REM
200 REM pointeur derniere
210 REM adresse libre dans
220 REM la mini-memoire
230 REM
240 CALL LOAD(28702,127,208)
250 REM
260 REM initialisation de
270 REM la table des
280 REM references externes
290 REM
300 DATA 73,77,65,71,69,32,248,248,80,79,
73,78,84,32,248,164,65,88,69,83,32,32,2
48,88
310 DATA 69,70,70,67,32,32,248,72,69,70,7
0,80,32,32,248,54,69,70,70,84,32,32,248
,50
320 FOR B=32720 TO 32767
330 READ V
340 CALL LOAD(B,V)
350 NEXT B
360 REM
370 REM chargement du code
380 REM objet dans
390 REM l'extension 32K
400 REM
410 DATA 225,129,2,128,161,129,6,130,255,
131,3,132,54,133,7,134,17,135,0,0,0,128
,224
420 DATA 129,0,130,12,131,0,132,6,133,0,1
34,7,135,0,0,15,0,7,2,16,1,4,194,2,0,22
4,10
430 DATA 4,240,2,128,248,10,22,252,192,13
0,19,6,2,0,38,118,4,240,2,128,62,118,22
,252
440 DATA 4,96,250,12,200,11,160,8,4,200,2
11,32,160,7,9,140,2,140,0,191,21,4,2,9,
0,191
450 DATA 98,76,16,1,4,201,192,8,192,73,6,
160,248,190,5,136,2,136,1,0,22,248,4,20
1,210
460 DATA 32,160,6,9,136,192,8,192,73,6,16
0,248,190,5,137,2,137,0,192,22,248,194,
224
470 DATA 160,8,4,96,250,12,7,7,208,32,160
,0,9,128,208,160,160,1,9,130,2,130,0,19
1,21
480 DATA 30,2,1,0,191,96,66,4,199,193,1,1
0,84,225,1,2,68,255,7,161,0,2,64,0,7,97
,0,2
490 DATA 3,128,0,11,3,249,3,224,10,209,96
,160,2,4,198,209,160,160,3,11,198,241,7
0,217
500 DATA 5,38,118,193,199,22,1,4,91,4,96,
250,12,216,32,160,4,248,26,2,1,160,10,4
,192
510 DATA 2,2,224,10,216,0,140,2,16,0,216,
0,140,2,16,0,220,96,136,0,128,129,22,25
2,216
520 DATA 32,248,14,131,212,2,1,248,12,192
,177,19,6,216,2,140,2,6,194,216,2,140,2
,16
530 DATA 248,2,0,0,91,2,1,208,0,216,0,140
,2,6,192,216,0,140,2,16,0,216,1,140,0,4
,193
540 DATA 2,2,3,0,2,0,0,88,216,0,140,2,6,1
92,216,0,140,2,6,193,216,1,140,0,6,193,
5,129
550 DATA 128,129,22,249,2,0,0,64,2,1,224,
10,2,2,248,10,216,0,140,2,6,192,216,0,1
40,2
560 DATA 16,0,216,49,140,0,128,129,22,252
,2,0,0,96,2,1,38,118,2,2,62,118,216,0,1
40,2
570 DATA 6,192,216,0,140,2,16,0,216,49,14
0,0,128,129,22,252,216,32,248,10,131,21
2,192
580 DATA 32,248,10,216,0,140,2,6,192,216,
0,140,2,2,224,131,224,6,160,0,14,152,32
,131
590 DATA 117,248,48,22,250,2,224,112,184,
2,1,248,30,192,177,19,6,216,2,140,2,6,1
94,216
600 DATA 2,140,2,16,248,2,0,0,64,2,1,160,
10,2,2,224,10,216,0,140,2,6,192,216,0,1
40,2
610 DATA 16,0,216,49,140,0,128,129,22,252
,4,192,216,0,131,124,4,91
620 FOR B=-2038 TO -1517
630 READ V
640 CALL LOAD(B,V)
650 NEXT B
660 END
```

Sous-programmes en Assembleur

Jean-luc Bazanegue

Nous avons, dans le numéro 3, analysé les différents modes d'adressage disponibles en Assembleur. Cette fois, nous allons travailler sur quelque chose de plus concret : les routines graphiques présentées dans ce numéro.

EFFT, EFFP et EFFC

Ces routines sont constituées par deux segments de programmes. Le premier place la valeur 0 dans les 6 Ko représentant la table des points. Le second réalise la même opération mais, cette fois, dans le tampon de 6 Ko situé dans la partie basse de l'extension 32 Ko. Cette zone de mémoire est destinée à recevoir les informations liées aux couleurs des points.

Puisque nous avons trois points d'entrée et seulement deux segments de programme, il faut utiliser un drapeau (le registre 2) qui nous permettra de savoir si nous avons abordé la routine par "EFFT" ou "EFFP".

Si, depuis le Basic, nous appelons la routine "EFFT", l'instruction SETO place à 1 les seize bits du registre 2. Si nous appelons la routine "EFFP", l'instruction CLR produira l'effet inverse, soit la mise à zéro des seize bits du registre 2. Ainsi, lorsque le programme aura terminé la réinitialisation du tampon "TIMAGE", l'instruction MOV R2,R2 testera l'état de R2 (elle place le bit "égal" du registre d'état à 1 si R2 est égal à zéro). Si R2 est égal à zéro, le programme retourne au Basic, sinon, le tableau "TCOUL" est réinitialisé.

Lors d'un appel de la routine "EFFC", le drapeau n'est pas nécessaire puisque le segment de programme se termine par un branchement vers la routine de retour au Basic.

AXES

La routine "AXES" utilise le registre 11, qui est aussi utilisé pour le retour au Basic. C'est pourquoi nous sommes obligé de sauvegarder le contenu de ce registre dans l'adresse @REG11.

Axe horizontal

Nous commençons par transférer le contenu de l'adresse −24569 dans le registre 12. Comme l'instruction MOVB copie la valeur dans l'octet de poids fort du registre, elle doit être reportée dans l'octet de poids faible. Pour cela, nous avons utilisé l'instruction SRL R12,8 (décalage logique du registre 12, de 8 positions vers la droite) qui place des zéros dans l'octet de poids fort de R12 tout en nous permettant d'obtenir une valeur comprise entre 0 et 255.

Notre écran contient 192 lignes (0 à 191) et un octet peut représenter une valeur comprise entre 0 et 255, il faut donc s'assurer de la validité du paramètre, étant donné qu'une valeur supérieure à 191 provoquerait une écriture à l'extérieur des tampons, donc une destruction partielle des routines, suivie d'un "plantage" à très court terme.

Pour éviter ce genre de surprises, la routine prend 191 comme valeur par défaut si le numéro de ligne est supérieur à 191.

La ligne 0 correspond à la première ligne de l'écran en partant du haut. Pour obtenir un format plus logique (ligne 0 en bas), la valeur représentant la position de l'axe horizontal est inversée.

Une fois cette opération effectuée, les points formant l'axe sont placés dans le tampon "TIMAGE" par la routine "POINT".

Axe Vertical

Mis à part le contrôle de validité et l'inversion de valeur, inutiles pour l'axe vertical, ce segment de programme fonctionne de la même façon que pour l'axe horizontal.

Avant de revenir au Basic, il faut placer l'adresse de retour dans le registre 11. Puisque nous l'avions sauvegardée dans l'adresse @REG11, l'instruction MOV @REG11,R11 réalise cette tâche très simplement.

POINT

La routine "POINT" peut être appelée à partir du Basic ou de la routine "AXES", nous avons donc utilisé, comme pour les routines "EFFT" et "EFFP", un drapeau permettant de savoir par quel point la routine a été abordée.

Si, lorsque la routine est appelée du Basic, la valeur de Y est supérieure à 191 (CI R2,191), le point à afficher est ignoré et la routine effectue un retour immédiat au programme appelant. Si la valeur est inférieure ou égale à 191, elle est inversée, tout comme dans la routine "AXES".

Le segment de programme qui calcule l'octet et le bit à modifier dans le tampon "TIMAGE" est celui que l'on retrouve dans tous les logiciels utilisant le mode Bit-map, il est donc inutile d'insister sur ce point.

IMAGE

Dans cette routine, les seuls points délicats sont les lectures et écritures en mémoire vive vidéo, sans utiliser VSBR, VMBR, VSBW, VMBW et VWTR. Nous avons préféré cette solution un peu plus complexe à mettre en oeuvre, mais qui permet de diminuer le temps d'exécution de la routine d'environ 50%.

Ecriture d'un octet en mémoire vive vidéo

Pour écrire un octet, il faut :

- placer à 1 le bit 1 de l'adresse dans laquelle vous désirez écrire. Par exemple, si vous voulez écrire un octet à l'adresse >2000 (00100000 00000000), cette adresse devient >6000 (01100000 00000000);
- transférer l'octet de poids faible de l'adresse dans laquelle on veut écrire dans l'adresse >8C02;
- provoquer un délai avec, par exemple, SWPB ou NOP (JMP $+1);
- transférer l'octet de poids fort de l'adresse dans laquelle on veut écrire dans l'adresse >8C02;
- provoquer à nouveau un délai;
- transférer la valeur à écrire dans l'adresse >8C00.

Lecture d'un octet en mémoire vive vidéo

La méthode est la même que pour l'écriture, sauf pour les points suivants :

- le bit 1 de l'adresse ne doit pas être placé à 1;
- au lieu de transférer une valeur dans l'adresse >8C00, il faut lire une valeur à l'adresse >8800

Modification d'un registre vidéo

Pour modifier un registre, il faut :

- transférer l'octet contenant le numéro du registre dans l'adresse >8C02. Le bit 0 de l'octet doit être placé à 1. Par exemple, pour modifier le registre >7 (00000111), la valeur >87 (10000111) doit être transférée;
- provoquer un délai;
- transférer l'octet contenant la valeur à charger dans le registre vers l'adresse >8C02.

Lecture ou écriture de plusieurs octets en mémoire vive vidéo

L'adresse >8C02 étant auto-incrémentée, il n'est pas nécessaire de donner à chaque fois l'adresse dans laquelle on désire lire ou écrire. Par exemple, si une valeur a été écrite dans l'adresse >0001, l'adresse >0002 est automatiquement pointée.

Il est ainsi possible de modifier ou de lire jusqu'à un maximum de 16 Ko en indiquant une seule adresse.

■

```
*****************************************
*                                       *
* ROUTINES GRAPHIQUES - HAUTE RESOLUTION *
*                                       *
*****************************************
* Copyright "99 Magazine", Gerard Santraille *
* et Jean-Luc Bazanegue                 *
*****************************************

       DEF  EFFT,EFFP,EFFC,AXES,POINT,IMAGE  * References externes.

       AORG >A000  * Origine absolue.

COORD  DATA 0      * Coordonnees  - X dans -24576  -  Y dans -24575.
COULP  DATA 0      * Couleurs - points "off" dans -24574
*                  *          - points "on" dans -24573.
COULE  DATA 0      * Couleur de l'ecran dans -24572.
POSAXE DATA 0      * Positions des axes -  Y dans -24570  -  X dans -24569.
REG11  DATA 0      * 2 octets pour la sauvegarde du registre 11 (retour Basic).
TBASIC BSS  >4000  * Tampon pour sauvegarde de la memoire vive video (16K).
TIMAGE BSS  >1800  * Tampon pour la creation de l'image en memoire vive (4K).
RVDP1  DATA >E181  * Registre VDP 1 en mode BIT-MAP (affichage inhibe).

REGVDP DATA >0280,>A181,>0682,>FF83     * Registres VDP pour le mode BIT-MAP.
       DATA >0384,>3685,>0786,>1187,0   *
RVDPBA DATA >0080,>E081,>0082,>0C83     * Registres VDP pour le Basic
       DATA >0084,>0685,>0086,>0787,0   * (valeurs par defaut).

TCOUL  EQU  >2676  * Tampon pour la determination des couleurs (4K).
ETAT   EQU  >837C  * Adresse du registre d'etat de l'interpreteur GPL.
RCLAV  EQU  >8375  * Adresse contenant le code de la touche pressee.
BACK   BYTE 15     * Code de la touche "BACK" (FCTN-9).
       EVEN        * Pour continuer l'assemblage sur une adresse paire.

***********************************************************
*                                                         *
* Routine "EFFT" : efface les tampons "TIMAGE" et "TCOUL". *
*    "    "EFFP" : efface le tampon "TIMAGE".              *
*    "    "EFFC" : efface le tampon "TCOUL".               *
* Appel du Basic par : - CALL LINK("EFFT").                *
*                      - CALL LINK("EFFP").                *
*                      - CALL LINK("EFFC").                *
*                                                         *
***********************************************************

EFFT   SETO R2             * R2 = drapeau haut si appel par CALL LINK("EFFT").
       JMP  EFFP2          * Saut vers effacement des points.
EFFP   CLR  R2             * R2 = drapeau bas si appel par CALL LINK("EFFP").
EFFP2  LI   R0,TIMAGE      * Charge l'adresse du tampon "TIMAGE" dans R0.
EFFP1  CLR  *R0+           * Met a zero le mot dont l'adresse est contenue
*                          * par le registre R0. Incremente R0 par 2.
       CI   R0,TIMAGE+>1800 * Test si fin de tampon.
       JNE  EFFP1          * Si non, poursuivre la mise a zero.
       MOV  R2,R2          * Controle de l'etat du drapeau.
       JEQ  FEFF           * Si bas, retour au Basic.
EFFC   LI   R0,TCOUL       * Charge l'adresse du tampon "TCOUL" dans R0.
EFFC1  CLR  *R0+           * Met a zero le mot dont l'adresse est contenue
*                          * par le registre R0. Incremente R0 par 2.
       CI   R0,TCOUL+>1800 * Test si fin de tampon.
       JNE  EFFC1          * Si non, poursuivre la mise a zero.
FEFF   B    @RETOUR        * Vers routine de retour au Basic.

*************************************************************************
*                                                                       *
* La routine "AXES" realise le trace des axes en fonction des adresses  *
* -24570 : axe vertical (Y). -24569 : axe horizontal (X).               *
* Couleurs des axes: - points "off" dans -24574                         *
*                    - points "on" dans -24573.                         *
*                                                                       *
*************************************************************************

AXES   MOV  R11,@REG11     * Sauvegarde de l'adresse de retour au Basic.

*******************
* Axe horizontal *
*******************
```

```
        CLR  R8                 * Zero dans R8.
        MOVB @POSAXE+1,R12      * Position dans l'octet de poids fort de R12.
        SRL  R12,8              * Decalage logique vers la droite de 8 positions.
        CI   R12,191            * Compare le contenu de R12 avec 191.
        JGT  AXEY1              * Si superieur, saut a AXEY1.
        LI   R9,191             * Sinon, charge la valeur 191 dans R9.
        S    R12,R9             * Soustrait R12 a R9. La position reelle est dans R9.
        JMP  AXEY               * Vers trace ligne horizontale.
AXEY1   CLR  R9                 * Ligne horizontale = 1ere ligne en bas de l'ecran.
AXEY    MOV  R8,R0              * Copie de R8 dans R0.
        MOV  R9,R1              * Copie de R9 dans R1.
        BL   @POINT2            * Appel de la routine pour calculer un point.
        INC  R8                 * Point suivant.
        CI   R8,256             * Verification si tous les points sont calcules.
        JNE  AXEY               * Si non, continuer.

****************
* Axe vertical *
****************

        CLR  R9                 * Zero dans R9.
        MOVB @POSAXE,R8         * Position dans l'octet de poids fort de R8.
        SRL  R8,8               * Decalage logique vers la droite de 8 positions.
AXEX    MOV  R8,R0              * Copie de R8 dans R0.
        MOV  R9,R1              * Copie de R9 dans R1.
        BL   @POINT2            * Appel de la routine pour calculer un point.
        INC  R9                 * Point suivant.
        CI   R9,192             * Verification si tous les points sont calcules.
        JNE  AXEX               * Si non, continuer.

        MOV  @REG11,R11         * Place l'adresse de retour au Basic dans R11.
        B    @RETOUR            * Branchement sur la routine de retour au Basic.

*****************************************************************************
*                                                                           *
* La routine "POINT" calcule l'octet et le bit a modifier dans le tampon   *
* "TIMAGE" en fonction des coordonnees X et Y transmises par le            *
* programme Basic aux adresses >A000 et >A001.                             *
* Les couleurs des points "off" et "on" doivent etre aux adresses          *
* -24574 et -24573.                                                        *
* appel a partir du Basic par CALL LINK("POINT").                          *
*                                                                           *
*****************************************************************************

POINT   SETO R7             * R7 = drapeau haut si appel a partir du Basic.
        MOVB @COORD,R0      * Copie le contenu de l'adresse >A000 dans l'octet de
*                           * poids fort de R0.
        SRL  R0,8           * Decale le contenu de R0 de 8 positions vers la droite
        MOVB @COORD+1,R2    * Copie le contenu de l'adresse >A001 dans l'octet de
*                           * poids fort de R2.
        SRL  R2,8           * Decale le contenu de R2 de 8 positions vers la droite
        CI   R2,191         * Compare le contenu de R2 (Y) avec 191.
        JGT  FHPLOT         * Si Y est superieur a 191, retour au Basic.
        LI   R1,191         * Charge la valeur 191 dans le registre 1.
        S    R2,R1          * Soustrait la valeur contenu dans le registre 2 a
*                           * R1 pour inverser Y. Y se trouve maintenant dans R1.
POINT2  CLR  R7             * R7 = drapeau bas si appel a partir de "AXES".
        MOV  R1,R4          * Copie le contenu de R1 dans R4.
        SLA  R4,5           * Decale le registre 4 de 5 positions vers la gauche.
        SOC  R1,R4          * Force a 1 les bits de R4 qui correspondent aux bits
*                           * a 1 de R4.
        ANDI R4,>FF07       * Intersection logique entre le registre 4 et >FF07.
        A    R0,R4          * Ajoute R0 a R4.
        ANDI R0,7           * Intersection logique entre le registre 0 et 7.
        S    R0,R4          * Soustrait R0 a R4.
*                           * R4 contient l'adresse de l'octet a modifier, par
*                           * rapport a l'adresse de base du tampon "TIMAGE".
*                           * R0 contient la position du bit a modifier dans
*                           * l'octet obtenu par R4.
        LI   R3,>8000       * Charge la valeur >8000 (1000000000000000 en binaire
*                           * dans R3.
        SRC  R3,0           * Decalage circulaire vers la droite de R3 du nombre
*                           * de positions contenu dans les 4 bits de droite de R0.
        SOCB R3,@TIMAGE(R4)     * Force a 1 le bit correspondant au bit a 1
*                               * du registre 3.
        MOVB @COULP,R5          * Couleur des points "off" dans R5.
        CLR  R6                 * Registre 6 = 0.
        MOVB @COULP+1,R6        * Couleur des points "on" dans R6.
        SRC  R6,12              * Decalage circulaire de 12 positions a droite.
        SOCB R6,R5              * Place a 1 les bits de R5 correspondants aux
*                               * bits a 1 de R6.
        MOVB R5,@TCOUL(R4)      * Transfere le contenu de l'octet de poids fort de
*                               * R5 dans l'octet situe a l'adresse @TCOUL +
*                               * la valeur contenue par R4.
```

```
       MOV  R7,R7              * Test de l'etat du drapeau.
       JNE  FHPLOT             * Si haut, retour au Basic.
       RT                      * Sinon, retour a la routine "AXES".
FHPLOT B    @RETOUR            * Branchement sur la routine de retour au Basic

*****************************************************************************
*                                                                           *
* La routine "IMAGE" sauvegarde le contenu de la memoire vive video dans le *
* tampon de 16K baptise  "TBASIC", initialise les registres VDP pour le     *
* mode BIT-MAP et affiche l'image en transferant le contenu du tampon       *
* "TIMAGE" dans la memoire vive video, a partir de l'adresse >0000, et le   *
* tampon "TCOUL" a partir de l'adresse >2000.                               *
* Appel a partir du Basic par CALL LINK("IMAGE")                            *
*                                                                           *
*****************************************************************************

IMAGE  MOVB @COULE,@REGVDP+14  * Copie de l'adresse -24572 (couleur de l'ecran).

****************************************
* Sauvegarde de la memoire vive video *
****************************************

       LI   R1,TBASIC          * Adresse du tampon en memoire vive du CPU.
       CLR  R0                 * Adresse du tampon en memoire vive video.
       LI   R2,TBASIC+>4000    * Nombre d'octet a transferer.
       MOVB R0,@>8C02      *
       NOP                  *
       MOVB R0,@>8C02       *
       NOP                     * Remplace la routine VMBR.
SB     MOVB @>8800,*R1+     *
       C    R1,R2           *
       JNE  SB             *

************************************************************
* Initialisation des registres video pour le mode BIT-MAP *
************************************************************

       MOVB @REGVDP+2,@>83D4   * Copie du registre VDP 1 dans >83D4.
*                              * (affichage inhibe).
       LI   R1,REGVDP          * Adresse des valeurs pour les registres video.
IRBM   MOV  *R1+,R2        *
       JEQ  ITLUT           *
       MOVB R2,@>8C02       * Remplace la routine VWTR.
       SWPB R2              *
       MOVB R2,@>8C02       *
       JMP  IRBM           *

***********************************************************************
* Initialisation table des lutins (pour eviter les images "fantomes")*
***********************************************************************

ITLUT  LI   R0,>005B       * Adresse en memoire vive video.
       LI   R1,>D000       * D0 = valeur a ecrire.
       MOVB R0,@>8C02  *
       SWPB R0          *
       MOVB R0,@>8C02   * Remplace VSBW.
       NOP              *
       MOVB R1,@>8C00  *

*****************************************************************
* Initialisation de la table image ecran (>00 a >FF trois fois) *
*****************************************************************

       CLR  R1            *
       LI   R2,768         *
       LI   R0,>0058       *
       MOVB R0,@>8C02      *
       SWPB R0             *
       MOVB R0,@>8C02       * Remplace VSBW.
ITE    SWPB R1             * Voir article pour explications.
       MOVB R1,@>8C00      *
       SWPB R1             *
       INC  R1             *
       C    R1,R2          *
       JNE  ITE           *

*******************************************************
* Transfert du tampon "TIMAGE" en memoire vive video *
*******************************************************

       LI   R0,>0040           * Adresse de la table des definitions.
       LI   R1,TIMAGE          * Adresse du tampon en memoire vive du CPU.
       LI   R2.TIMAGE+>1800    * Derniere adresse du tampon.
```

```
        MOVB R0,@>8C02         *
        SWPB R0                 *
        MOVB R0,@>8C02          *
        NOP                      * Remplace VMBR. (voir article).
TTI     MOVB *R1+,@>8C00        *
        C    R1,R2              *
        JNE  TTI               *

*************************************************************
* Transfert du tampon "TCOUL" en memoire vive video *
*************************************************************

        LI   R0,>0060          *
        LI   R1,TCOUL           *
        LI   R2,TCOUL+>1800     *
        MOVB R0,@>8C02          *
        SWPB R0                  * Meme methode que pour
        MOVB R0,@>8C02           * le tampon "IMAGE".
        NOP                     *
TTC     MOVB *R1+,@>8C00        *
        C    R1,R2              *
        JNE  TTC               *

**********************************
* Retablissement de l'affichage *
**********************************

        MOVB @RVDP1,@>83D4
        MOV  @RVDP1,R0
        MOVB R0,@>8C02
        SWPB R0
        MOVB R0,@>8C02

*************************
* Scrutation du clavier *
*************************

        LWPI >83E0             * Memoire de travail de l'interpreteur GPL.
CLAV    BL   @>000E
        CB   @RCLAV,@BACK      * Controle si la touche "BACK" est pressee.
        JNE  CLAV              * Si non, continue la scrutation.
        LWPI >20BA             * Charge la memoire de travail utilisateur (USRWSP).
*                              * NOTE: Pour une utilisation de cette routine avec le
*                              * module Mini-Memoire, il faut remplacer la
*                              * valeur >20BA par >70B8.

****************************************************************************
* Reinitialisation des registres video pour le Basic (valeurs par defaut)*
****************************************************************************


        LI   R1,RVDPBA        *
IRB     MOV  *R1+,R2           *
        JEQ  TTBAS             *
        MOVB R2,@>8C02          * Meme methode que pour le mode BIT-MAP.
        SWPB R2                *
        MOVB R2,@>8C02         *
        JMP  IRB              *

*******************************************************************
* Memoire vive video comme avant l'appel de la routine "IMAGE" *
*******************************************************************

TTBAS   LI   R0,>0040            * Adresse du tampon en memoire vive video.
        LI   R1,TBASIC           * Adresse du tampon en memoire vive du CPU.
        LI   R2,TBASIC+>4000     * Derniere adresse du tampon.
        MOVB R0,@>8C02         *
        SWPB R0                 *
        MOVB R0,@>8C02          *
        NOP                      * Remplace  VMBW.
TTB     MOVB *R1+,@>8C00        *
        C    R1,R2              *
        JNE  TTB               *

******************************
* Retour au programme Basic *
******************************

RETOUR  CLR  R0               * Zero dans le registre 0.
        MOVB R0,@ETAT         * Zero dans le registre d'etat de l'interpreteur GPL.
        RT                    * Equivalent a : B *R11.
        END
```

Horloge

Christian Monfort

Ce programme, écrit en **Basic Etendu**, transforme l'écran de votre TI en horloge - chronomètre - calendrier.

Le démarrage de l'horloge se fait après avoir entré la date (en chiffres).

Les aiguilles sont remplacées par :

- la coloration en bleu du chiffre indiquant l'heure en cours;
- une grosse étoile rouge indiquant les minutes;
- une petite étoile blanche qui tourne au rythme des seconde.

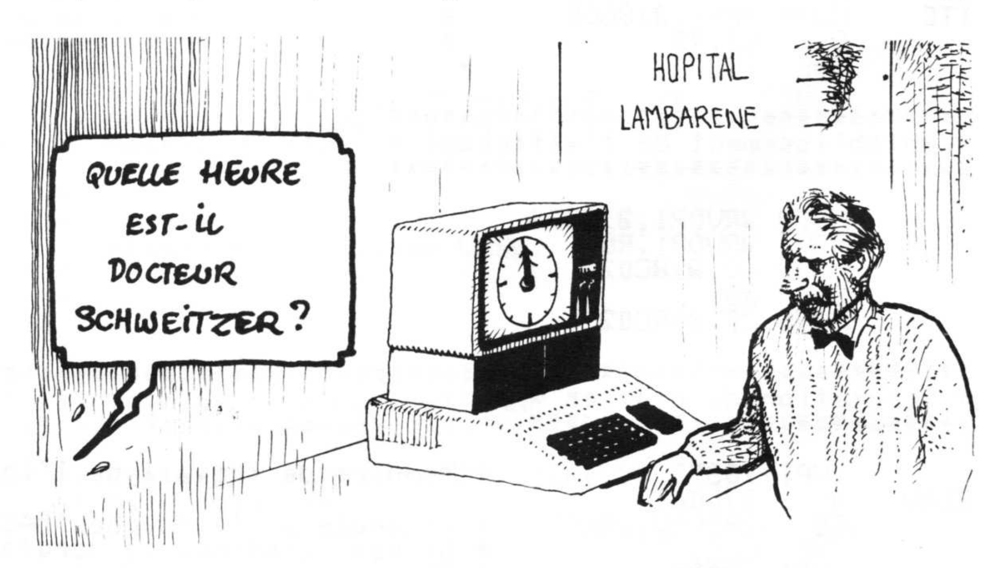
Le chronomètre démarre, s'arrête et s'efface en appuyant sur une touche quelconque. Ces opérations n'étant prises en compte qu'au passage sur une instruction CALL KEY, il faut maintenir le doigt sur la touche jusqu'à ce que l'on entende le signal sonore.

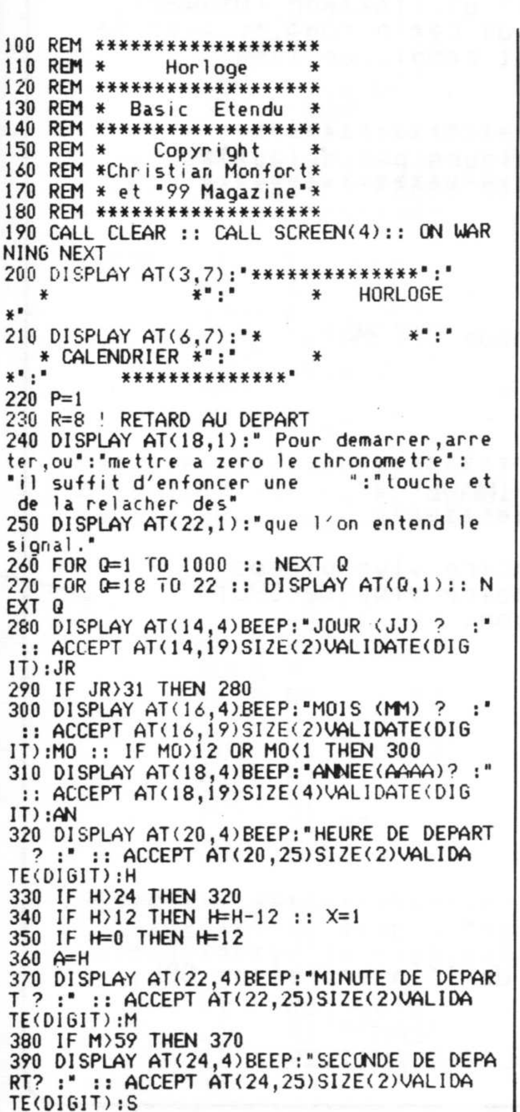
Le calendrier change de date lorsque l'horloge marque 24 heures, ce qui est la moindre des choses. Les années bissextiles sont prévues.

Cette horloge n'est pas d'une grande utilité. Elle montre (sans jeu de mots) cependant la façon de faire tourner non pas des aiguilles, mais des points les matérialisant en fonction du temps.

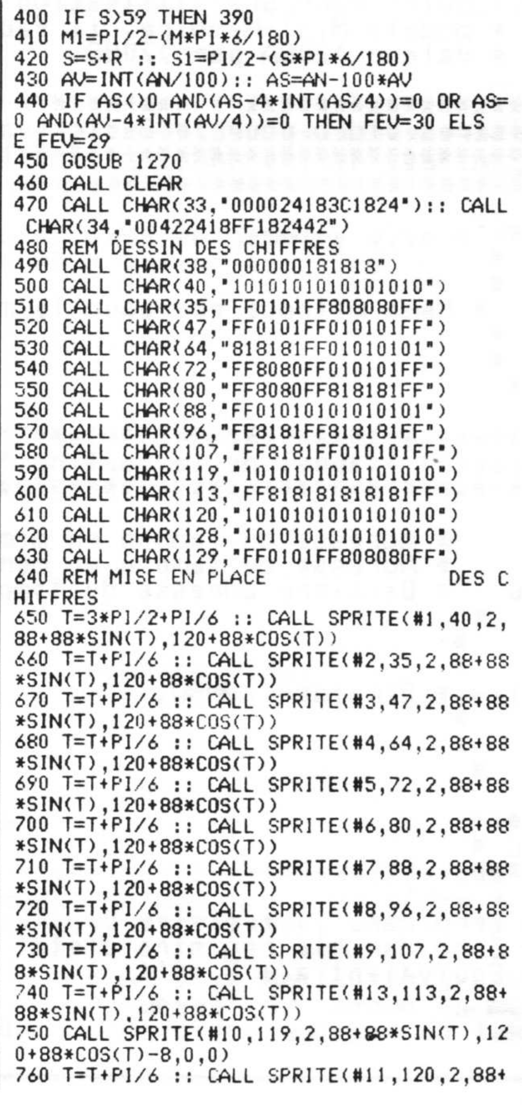
Note : si vous utilisez une extension 32K, il est possible que l'horloge prenne rapidement de l'avance. Dans ce cas, vous devrez modifier les lignes 1000 et 1010 en conséquence (variable DELAI).

***NDLR** : comme prévu, Christian Monfort se voit attribuer un abonnement pour quatre numéros, avec cassettes.* ■





```
100 REM *******************
110 REM *     Horloge     *
120 REM *******************
130 REM *  Basic  Etendu  *
140 REM *******************
150 REM *    Copyright    *
160 REM *Christian Monfort*
170 REM * et "99 Magazine"*
180 REM *******************
190 CALL CLEAR :: CALL SCREEN(4):: ON WAR
NING NEXT
200 DISPLAY AT(3,7):"***************":"
  *             *":"      *   HORLOGE
*"
210 DISPLAY AT(6,7):"*             *":"
  * CALENDRIER *":"            *
*":"      ***************"
220 P=1
230 R=8 ! RETARD AU DEPART
240 DISPLAY AT(18,1):" Pour demarrer,arre
ter,ou":"mettre a zero le chronometre":
"il suffit d'enfoncer une     ":"touche et
 de la relacher des"
250 DISPLAY AT(22,1):"que l'on entend le
signal."
260 FOR Q=1 TO 1000 :: NEXT Q
270 FOR Q=18 TO 22 :: DISPLAY AT(Q,1):: N
EXT Q
280 DISPLAY AT(14,4)BEEP:"JOUR (JJ) ?  :"
 :: ACCEPT AT(14,19)SIZE(2)VALIDATE(DIG
IT):JR
290 IF JR>31 THEN 280
300 DISPLAY AT(16,4)BEEP:"MOIS (MM) ?  :"
 :: ACCEPT AT(16,19)SIZE(2)VALIDATE(DIG
IT):MO :: IF MO>12 OR MO<1 THEN 300
310 DISPLAY AT(18,4)BEEP:"ANNEE(AAAA)? :"
 :: ACCEPT AT(18,19)SIZE(4)VALIDATE(DIG
IT):AN
320 DISPLAY AT(20,4)BEEP:"HEURE DE DEPART
  ? :" :: ACCEPT AT(20,25)SIZE(2)VALIDA
TE(DIGIT):H
330 IF H>24 THEN 320
340 IF H>12 THEN H=H-12 :: X=1
350 IF H=0 THEN H=12
360 A=H
370 DISPLAY AT(22,4)BEEP:"MINUTE DE DEPAR
T ? :" :: ACCEPT AT(22,25)SIZE(2)VALIDA
TE(DIGIT):M
380 IF M>59 THEN 370
390 DISPLAY AT(24,4)BEEP:"SECONDE DE DEPA
RT? :" :: ACCEPT AT(24,25)SIZE(2)VALIDA
TE(DIGIT):S
```



```
400 IF S>59 THEN 390
410 M1=PI/2-(M*PI*6/180)
420 S=S+R :: S1=PI/2-(S*PI*6/180)
430 AV=INT(AN/100):: AS=AN-100*AV
440 IF AS<>0 AND(AS-4*INT(AS/4))=0 OR AS=
0 AND(AV-4*INT(AV/4))=0 THEN FEV=30 ELS
E FEV=29
450 GOSUB 1270
460 CALL CLEAR
470 CALL CHAR(33,"000024183C1824"):: CALL
 CHAR(34,"00422418FF182442")
480 REM DESSIN DES CHIFFRES
490 CALL CHAR(38,"000000181818")
500 CALL CHAR(40,"1010101010101010")
510 CALL CHAR(35,"FF0101FF808080FF")
520 CALL CHAR(47,"FF0101FF010101FF")
530 CALL CHAR(64,"818181FF01010101")
540 CALL CHAR(72,"FF8080FF010101FF")
550 CALL CHAR(80,"FF8080FF818181FF")
560 CALL CHAR(88,"FF01010101010101")
570 CALL CHAR(96,"FF8181FF818181FF")
580 CALL CHAR(107,"FF8181FF010101FF")
590 CALL CHAR(119,"1010101010101010")
600 CALL CHAR(113,"FF818181818181FF")
610 CALL CHAR(120,"1010101010101010")
620 CALL CHAR(128,"1010101010101010")
630 CALL CHAR(129,"FF0101FF808080FF")
640 REM MISE EN PLACE                DES C
HIFFRES
650 T=3*PI/2+PI/6 :: CALL SPRITE(#1,40,2,
88+88*SIN(T),120+88*COS(T))
660 T=T+PI/6 :: CALL SPRITE(#2,35,2,88+88
*SIN(T),120+88*COS(T))
670 T=T+PI/6 :: CALL SPRITE(#3,47,2,88+88
*SIN(T),120+88*COS(T))
680 T=T+PI/6 :: CALL SPRITE(#4,64,2,88+88
*SIN(T),120+88*COS(T))
690 T=T+PI/6 :: CALL SPRITE(#5,72,2,88+88
*SIN(T),120+88*COS(T))
700 T=T+PI/6 :: CALL SPRITE(#6,80,2,88+88
*SIN(T),120+88*COS(T))
710 T=T+PI/6 :: CALL SPRITE(#7,88,2,88+88
*SIN(T),120+88*COS(T))
720 T=T+PI/6 :: CALL SPRITE(#8,96,2,88+88
*SIN(T),120+88*COS(T))
730 T=T+PI/6 :: CALL SPRITE(#9,107,2,88+8
8*SIN(T),120+88*COS(T))
740 T=T+PI/6 :: CALL SPRITE(#13,113,2,88+
88*SIN(T),120+88*COS(T))
750 CALL SPRITE(#10,119,2,88+88*SIN(T),12
0+88*COS(T)-8,0,0)
760 T=T+PI/6 :: CALL SPRITE(#11,120,2,88+
```

```
88*SIN(T),120+88*COS(T),0,0)
770 CALL SPRITE(#14,120,2,88+88*SIN(T),12
0+88*COS(T)-8)
780 T=T+PI/6 :: CALL SPRITE(#15,129,2,92+
88*SIN(T),120+88*COS(T))
.790 CALL SPRITE(#12,128,2,92+88*SIN(T),12
0+88*COS(T)-8)
800 CALL COLOR(#A,5):: IF H>9 THEN CALL C
OLOR(#A+3,5)
810 CALL SPRITE(#16,33,16,88-72*SIN(S1),1
20+72*COS(S1))
820 CALL SPRITE(#17,34,14,88-64*SIN(M1),1
20+64*COS(M1))
830 W=18
840 FOR T1=11*PI/6 TO 0 STEP -PI/6 :: CAL
L SPRITE(#W,38,13,88-72*SIN(T1),120+72*
COS(T1)):: W=W+1 :: IF W>28 THEN 860
850 NEXT T1
860 CALL HCHAR(12,25,38):: CALL COLOR(1,1
3,1)
870 IF JR<10 THEN JR$="0"&STR$(JR)ELSE JR
$=STR$(JR)
880 DISPLAY AT(14,11):JOUR$ :: DISPLAY AT
(15,14):JR$ :: DISPLAY AT(16,10):MOI$
890 FOR S2=S TO 59
900 REM CHRONOMETRE
910 CALL KEY(0,K,STATE)
920 IF STATE=0 THEN 940 ELSE CALL SOUND(1
00,110,0)
930 IF C+D<>0 THEN 960 ELSE 980
940 IF C+D=0 THEN 990
950 IF AR=1 THEN 1000 ELSE 980
960 IF AR<>1 THEN 970 ELSE AR=0 :: C=0 ::
 D=0 :: GOTO 1000
970 AR=1 :: GOTO 1000
980 C=C+1 :: IF C=60 THEN C=0 :: D=D+1
990 AR=0
1000 IF C+D=0 THEN DELAI=121 ELSE DELAI=9
2
1010 IF AR=1 THEN DELAI=98
1020 FOR DEL=1 TO DELAI :: NEXT DEL
1030 IF C+D<>0 THEN DISPLAY AT(10,14-LEN(
STR$(D)))SIZE(5):STR$(D);".";STR$(C)ELS
E DISPLAY AT(10,12)
1040 S1=S1-(PI/30)
1050 CALL LOCATE(#16,88-72*SIN(S1),120+72
*COS(S1))
1060 NEXT S2 :: S=0 :: S1=PI/2
1070 M1=M1-PI/30
1080 CALL LOCATE(#17,88-64*SIN(M1),120+64
*COS(M1))
1090 M=M+1
```

```
1100 IF M=60 THEN M=0 ELSE 870
1110 REM COULEUR DES CHIFFRES
1120 CALL SOUND(100,400,0,4000,2):: M1=PI
/2 :: A=A+1
1130 IF H<9 THEN B=A ELSE B=A+3
1140 CALL COLOR(#A-1,2):: IF H>9 THEN CAL
L COLOR(#B-1,2)
1150 IF A=13 THEN A,B=1
1160 IF A=12 AND X=1 THEN JR=JR+1 :: X=0
:: SA=SA+1
1170 IF SA=8 THEN SA=1
1180 JOUR$=SEG$(J$,8*SA-7,8)
1190 CALL COLOR(#A,5):: CALL COLOR(#B,5)
1200 REM CALENDRIER
1210 IF (MO=1 OR MO=3 OR MO=5 OR MO=7 OR
MO=8 OR MO=10)AND JR=32 THEN MO=MO+1 ::
 JR=1 :: GOTO 1250
1220 IF (MO=4 OR MO=6 OR MO=9 OR MO=11)AN
D JR=31 THEN MO=MO+1 :: JR=1 :: GOTO 12
50
1230 IF MO=2 AND JR=FEV THEN MO=MO+1 :: J
R=1 :: GOTO 1250
1240 IF MO=12 AND JR=32 THEN MO=1 :: JR=1
:: AN=AN+1
1250 MOI$=SEG$(M$,9*MO-8,9):: GOTO 870
1260 END
1270 ! CALCUL DU JOUR ET DU          MOIS
1280 J$=" lundi   mardi  mercredi jeudi
vendredi samedi dimanche"
1290 M$=" janvier  fevrier   mars    avr
il     mai      juin    juillet    aout
 septembre octobre novembre decembre "
1300 A1=INT(AN/100):: A2=AN-100*A1 :: N=0
1310 IF MO>2 THEN 1400
1320 N=2
1330 IF A2=0 THEN 1380
1340 R=A2-4*INT(A2/4)
1350 IF R<>0 THEN 1400
1360 N=1
1370 GOTO 1400
1380 R=A1-4*INT(A1/4)
1390 IF R=0 THEN N=1
1400 CC=INT(365.25*A2)+INT(30.56*MO)+N+JR
1410 SA=3+CC-7*INT((CC+2)/7)
1420 JOUR$=SEG$(J$,8*SA-7,8)
1430 MOI$=SEG$(M$,9*MO-8,9)
1440 RETURN
```

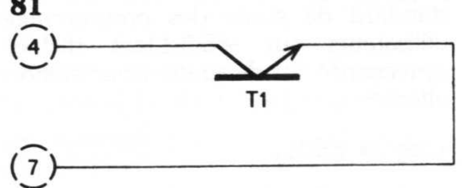
Un détecteur de lumière pour le TI-99

Guy Fix

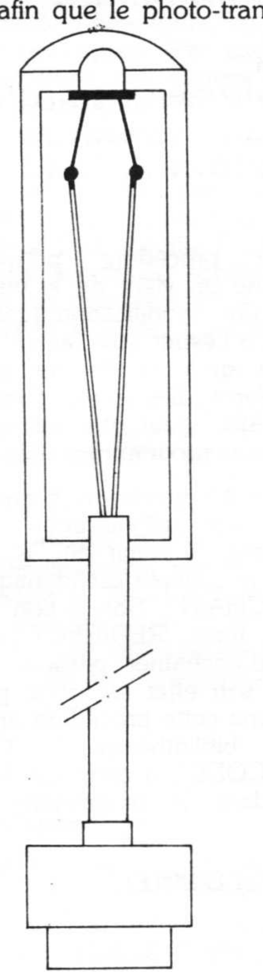
Ce détecteur utilise l'entrée des manettes de jeu. Il est possible de le brancher sur la manette A ou B ou, pourquoi pas, sur les deux entrées.
Le brochage et le branchement du détecteur sont donnés pour un déclenchement simulant le bouton de tir des manettes.

Manette A - 7 et 4
Manette B - 2 et 4

T1 - Photo-transistor genre TIL 81



Le montage peut être effectué dans un tube, afin que le photo-transistor ne soit pas affecté par la lumière ambiante.



- Prise CANON 9 points ou SOURIAU DEF-9S

Pour toutes les autres utilisations, le brochage de la prise est donné ci-dessous.

Prise chassis

1 2 3 4 5
6 7 8 9

1 - libre
2 - manette B
3 - haut
4 - bouton de tir
5 - gauche
6 - libre
7 - manette A
8 - bas
9 - droite

Exemple : pour un branchement simulant le déplacement de la manette B vers la droite, raccorder le détecteur sur 2 et 9.

Explication sommaire du fonctionnement

Si T1 est éclairé, il se sature et "relie" les points 4 et 7, modifiant ainsi l'état de la variable affectée au bouton de tir (code 18).

NDLR : cette petite réalisation constitue une première étape vers la réalisation d'un crayon optique. En fait, toute la partie "hardware" est résumée ci-dessus. Le reste est uniquement un problème de logiciel. En résumé : il faut écrire une routine en Assembleur (le Basic est beaucoup trop lent pour ce type d'application) capable de détecter la position du crayon par rapport à l'écran.
Philippe Massénat nous a aimablement proposé sa collaboration pour effectuer la collecte des informations concernant ce sujet. Si vous avez déjà travaillé sur le sujet, ou si vous avez trouvé la solution du problème, vous pouvez écrire à : Philippe Massénat - 33, Avenue Habert de Montmort - 78320 Le Mesnil-Saint-Denis. ■

Le club M.T.I. se présente

Nous avons le plaisir de vous faire connaître l'existence à Nice d'un club d'utilisateurs du TI-99/4A.

Ce club existe depuis le mois de juillet 1983 et regroupe des adhérents à travers toute la France. Il est organisé de façon à pouvoir fonctionner par correspondance. Les adhérents reçoivent un bulletin périodique, ainsi que deux programmes gratuits à choisir chaque trimestre dans une liste de programmes éditée dans le bulletin (seul le support cassette ou disquette est facturé). Une permanence téléphonique, réservée aux adhérents, est assurée de 8 heures à 12 heures, et de 14 heures à 18 heures.

Des accords passés avec des fournisseurs permettent aux adhérents de se procurer du matériel à prix réduit.

Le club est constitué sous la forme d'une association à but non lucratif. L'adresse est :
M.T.I. - 6, rue Georges Ville - 06300 NICE - Téléphone : (93) 26 48 48 ■

Le système P-UCSD

Gérard Santraille

La carte P-Code permet d'accéder au système P-UCSD, parfois appelé POS (Pascal Operating System). Ce système d'exploitation, même s'il est assez lourd, possède des capacités intéressantes.
Revenons très brièvement sur les commandes-système primaires du POS, qui apparaissent sur la ligne guide, lorsque le système est démarré.

- "I(nit" ré-initialise le système, et lance l'exécution de "SYSTEM.STARTUP", si ce fichier est présent sur le volume racine (nous reviendrons sur cette possibilité).
- "E(dit" charge l'éditeur (appelé "SYSTEM.EDITOR") et le fichier de travail, s'il existe (SYSTEM.WRK).
- "R(un" exécute le fichier de travail. Si seul "SYSTEM.WRK.TEXT" est présent sur la disquette ou si le fichier-code est une version antérieure du fichier-texte, le compilateur est appelé automatiquement. De même, si le fichier-code nécessite une édition de liens, le "linker" est chargé.
- "C(omp" charge le compilateur (appelé "SYSTEM.COMPILER") et opère directement sur le fichier de travail (s'il existe).
- "L(ink" charge l'éditeur de liens (appelé "SYSTEM.LINKER") et agit automatiquement sur le fichier de travail.
- "X(ecute" lance l'exécution d'un fichier-code. Cette commande permet également de spécifier certaines options modifiant l'environnement du système. On peut, par exemple, déclarer directement le préfixe par défaut, sans faire appel au "FILER". On peut aussi, et c'est vraisemblablement l'une des particularités les plus intéressantes du POS, réorienter les fichiers standards d'entrée/sortie du système ou d'un programme. Nous reviendrons largement sur cette option, car elle conditionne entièrement l'utilisation des fichiers moniteur.
- "A(ssem" charge l'assembleur et opère directement sur le fichier de travail s'il est présent.
- "F(ile" appelle le "FILER".
- "H(alt" est une commande supportée par la version IV.0, qui permet de sortir du P-SYSTEME; on revient ainsi à la mire d'introduction.
- "U(ser restart" permet de relancer directement le dernier programme exécuté, sans modifier les valeurs des différentes variables.
- "M(onitor" permet de débuter, de suspendre ou de terminer une session moniteur.

Intérêt et utilisation de "SYSTEM.STARTUP"

Lorsque le système est démarré ou ré-initialisé, le fichier "SYSTEM.STARTUP" est automatiquement exécuté s'il est présent sur le volume racine.
"SYSTEM.STARTUP" peut être un fichier-code quelconque.

Cette possibilité est intéressante lorsqu'on désire orienter automatiquement le système (travail à distance à partir d'un autre terminal, en dirigeant les entrées/sorties du système sur REMIN: et REMOUT:) ou initialiser correctement une imprimante (par exemple, en exécutant l'utilitaire MODRS232) dès la mise en service, ou encore diriger les entrées du système sur un fichier moniteur. Pour ma part, sur la disquette contenant l'éditeur, j'ai toujours sous le nom de "SYSTEM.STARTUP" une version modifiée et automatique de l'utilitaire "MODRS232" qui initialise correctement mon imprimante (PIO).

Modification des fichiers d'entrée/sortie des programmes ou du système

Modification des entrées/sorties des programmes

Les programmes Pascal admettent deux fichiers standards (INPUT et OUTPUT) qu'il n'est pas nécessaire de définir. "INPUT" correspond au clavier et "OUTPUT" à l'écran. Il est possible de changer ces fichiers par défaut par la clause "PI=nom de fichier" (pour Program Input) et "PO=nom de fichier" (pour Program Output), accessible à partir de X(ecute du mode commande. Ainsi, presser X en mode commande provoque l'affichage de :

Xecute
what file?

Si on répond par "PO=printer:", toutes les impressions des programmes se feront, sauf spécification contraire, sur l'imprimante. Il est possible d'effectuer plusieurs modifications à la fois, en les séparant par un espace. Ainsi, "PO=printer: PI=remin:" affecte les fichiers d'entrée et de sortie des programmes utilisateurs.

On peut également modifier ces fichiers standards par un programme utilisateur, au moyen de la procédure "CHAIN", qui a le même effet que Xecute. La procédure "CHAIN" permet, comme son nom l'indique, d'enchaîner un programme à un autre (il s'agit en fait d'un overlay dynamique semblable au RUN "programme" du Basic Etendu). L'action de cette procédure n'est pas immédiate : la chaîne d'options d'exécution spécifiée est mise en attente et ne sera effectivement exécutée qu'à la fin du programme principal. Le programme suivant modifie le fichier standard de sortie des programmes utilisateurs sur #5:list.text (fichier sauvegardé sur disquette pour édition ultérieure).

```
Program ESSAI;

var .................
    .................

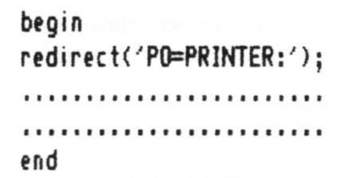
begin
  chain('PO=£5:LIST.TEXT');
  .....................
  .....................
end;
```

Cette procédure permet donc, comme on vient de le voir, d'effectuer une modification (gardons toujours à l'esprit que l'action n'est exécutée qu'à la fin du programme appelant). De façon plus générale, "CHAIN" peut être utilisé pour lier plusieurs programmes utilisateurs.

Une autre procédure permet d'effectuer une modification par programme : il s'agit de "REDIRECT", dont la syntaxe est identique à celle de "CHAIN". Son action est immédiate, mais "REDIRECT" ne permet pas d'enchaîner plusieurs programmes; son effet est donc plus limité. Comme cette procédure appartient à la bibliothèque "COMMANDIO.CODE", il convient de la déclarer dans le programme appelant. Ainsi :

```
Program EXEMPLE;

Uses COMMANDIO.CODE;
```



```
begin
redirect('PO=PRINTER:');
......................
......................
end
```

écrira sur l'imprimante chaque fois qu'il rencontrera un ordre d'impression ne désignant pas nommément le fichier destinataire. De la même façon, on peut spécifier le fichier d'entrée par défaut du système au moyen de "PI= nomdefichier".

Modifications des entrées/sorties du système

A la mise sous tension, lorsqu'on accède "naturellement" au P-SYSTEM, l'ordinateur communique avec l'utilisateur par le clavier et l'écran (ou le moniteur pour les mieux équipés), rien de plus normal jusque là. Il est cependant possible d'orienter le système sur l'interface RS232, sur l'imprimante, ou même sur un fichier d'une disquette. Cette modification s'effectue de la même façon que précédemment, c'est-à-dire au moyen des procédures "CHAIN" et "REDIRECT".

La syntaxe est :
CHAIN('I=REMIN: O=REMOUT:')

avec toujours I pour INPUT et O pour OUTPUT.
Comme pour les modifications des entrées/sorties des programmes, on peut spécifier plusieurs options dans le même appel à la procédure, en les séparant par un espace. Par exemple, une orientation des entrées du système sur un modem connecté à l'ordinateur, via l'interface RS232, permet de commander votre système à distance (ce n'est, hélas, pas tout à fait aussi simple que cela). De la même façon, si l'on spécifie l'imprimante comme fichier de sortie du système, tous les messages-guide seront imprimés et n'apparaîtront plus à l'écran.

Une possibilité intéressante consiste à diriger les entrées du système vers un fichier sur disquette. Ce fichier de type "TEXT" ou "DATA", qui peut être créé par l'éditeur de texte ou à partir d'une session moniteur, est alors lu séquentiellement par le système, et exécuté au fur et à mesure. Supposons que le fichier "DEMO.TEXT" soit situé sur le volume "GG1:", et qu'il contienne le texte suivant :

```
FD19
NYQE*LETTRE
/R/Monsieur//Madame/ jb
```

Si, en mode commande, on presse "X" et que l'on réponde "I=GG1:DEMO.TEXT" quand apparaît le prompteur "What File?", le système réagit comme si l'utilisateur pressait une à une les touches contenues dans le fichier "GG1:DEMO.TEXT".

Le "FILER" est chargé (F) et la date mise au 19 du mois courant (D19 puis passage à la ligne interprété comme un RETURN), le fichier de travail est ensuite détruit (N) et le Directory remis à jour (Y en réponse à Workfile removed update directory ?). Le système quitte ensuite le FILER (Q) et charge l'éditeur (E), qui charge à son tour le fichier "LETTRE.TEXT" du volume racine (*LETTRE). Rappelons que le volume racine (identifié par *) est le volume présent en "#4:", lorsqu'on accède au P-SYSTEM. Le système remplace ensuite toutes les chaînes 'Monsieur' par 'Madame' (/R/Monsieur//Madame/). L'espace entre Madame/ et jb permet de poursuivre l'exécution si la chaîne "Monsieur" n'a pas été trouvée (ERROR: Pattern not found. type<SP>); le système repositionne alors le curseur au début du texte (JB), puis rend la main à l'utilisateur (fin de fichier).

Comme on vient de le voir, le système peut effectuer automatiquement (et séquentiellement) toute une série de commandes et/ou de manipulations, à condition d'être dirigé vers un fichier adéquat. On peut créer un fichier moniteur de deux façons différentes :

1) à partir de l'éditeur de texte. D'un emploi très souple, le fichier obtenu pourra évidemment être réédité et corrigé. Le seul petit inconvénient de ce procédé est qu'un fichier créé par l'éditeur occupe au minimum 4 blocs (il mémorise en effet plusieurs autres informations qui ne nous intéressent pas) alors qu'un fichier moniteur d'un seul bloc est généralement suffisant;
2) en mémorisant une session moniteur. En mode commande, la pression sur M provoque l'affichage du prompteur. Monitor: B(egin E(nd A(bort puis S(uspend R(esume, chaque fonction permettant respectivement de débuter, finir, annuler, suspendre et reprendre une session moniteur.

Remarque : lors d'une session moniteur, il n'est pas possible de mémoriser la pression de M (qui permet justement d'avoir accès au moniteur).

Modification des entrées (système ou programmes) par chaîne (string)

Nous avons vu que lors d'une modification, qu'elle soit manuelle par le "X(ecute" du mode commande, ou automatique à partir d'un programme utilisateur par les procédures "CHAIN" ou "REDIRECT", on spécifiait toujours "I=nomdefichier" ou "PI=nomdefichier". Il est également possible d'écrire "I=chaîne" ou "PI=chaîne". Le système ou le programme s'oriente alors sur le CONTENU de la chaîne.

Ainsi, I="FD19,NYQ" charge le "FILER", met la date au 19 du mois courant, efface le fichier travail, met à jour le directoire, et retourne au mode commande. Cette écriture plus légère permet de ne pas accéder systématiquement à un fichier, lorsque les manipulations à effectuer ne sont pas trop nombreuses.
Remarque : dans une chaîne utilisée pour orienter les entrées, les virgules sont interprétées comme des retours-chariot.

Exemples d'application

Mise à jour automatique de la date à chaque session du P-SYSTEM

Si vous placez le programme suivant sur la disquette contenant l'éditeur, sous le nom "SYSTEM.STARTUP", chaque fois que l'on accède au P-system ou que l'on presse I en mode commande, le système demandera la date (à condition, bien entendu, que la disquette soit présente en #4).
Remarque : pressez simplement "ENTER" pour ne pas modifier la date.

```
Program DATE;

var action,date:string;

begin
write(chr(12),'date : ',chr(7));
readln(date);
if date=''
then
 exit(program)
else
 begin
  action:=concat('I="FD',date,',"');
  redirect(action)
 end
end.
```

Conversion automatique de caractères spéciaux

Le TI-99 ne possédant pas de caractères accentués, il est nécessaire de faire appel à un "traducteur" qui interprétera certains caractères (définis à l'avance par l'utilisateur), lors d'une édition sur imprimante. Le programme suivant convertit automatiquement un fichier de type "TEXT", comportant certains caractères spéciaux, en caractères accentués ou en commandes de l'imprimante (chacun devra l'adapter à sa configuration : il

s'agit ici d'une SEIKOSHA GP100). Rappelons au passage que l'éditeur de texte n'accepte pas de caractère de contrôle (ils sont systématiquement remplacés par un point d'interrogation), et qu'il est donc impossible de commander l'imprimante directement à partir d'un fichier "TEXT", lors de son transfert.

```
Program impression;

var
      imp,lect:interactive;
      car:char;
      asc:integer;
      fichier:string;

begin
  write(chr(7),' fichier : ');
  readln(fichier);
  fichier:=concat(fichier,'.text');
  reset(lect,fichier);
  rewrite(imp,'PRINTER:');
  while not eof(lect)do
    begin
      read(lect,car);
      if eoln(lect)
        then
          writeln(imp)
        else
          begin
            asc:=ord(car);
            case car of
              '{':
                  asc:=14;
              (double largeur)
              '}':
                  asc:=15;
              (simple largeur)
              '[':
                  asc:=166;
              (e accent aigu)
              ']':
                  asc:=167;
              (e accent grave)
              '~':
                  asc:=168;
              (e accent circonflexe)
              '@':
                  asc:=162;
              (a accent aigu)
              '`':
                  asc:=182;
              (u accent aigu)
              '|':
                  asc:=183;
              (u accent circonflexe)
              '^':
                  asc:=179;
              (o accent circonflexe)
            end;
            write(imp,chr(asc))
          end;
    end;
  writeln(imp)
end.
```

Ce programme est aisément adaptable ou extensible selon les besoins. Il suffit de modifier ou d'allonger la liste du :

```
case car of
...........
...........
end;
```

Lorqu'on désire visualiser le contenu du fichier à l'écran, il convient de redonner aux caractères spéciaux une forme plus sympathique, ou surtout plus intelligible. Deux façons de procéder :

1) On écrit un petit programme qui explore tout le fichier "TEXT", et qui remplace au fur et à mesure (lecture séquentielle) tous les caractères codés.
2) On fait exécuter le même travail par l'éditeur de texte, qui comporte une commande parfaitement adaptée à ce type de manipulation.

Ainsi, le programme suivant redirige les entrées du système sur une chaîne. Le système charge alors l'éditeur qui, au moyen de "REPLACE", effectue le travail demandé.

```
program DECODE;

(*$U #4:COMMANDIO.CODE *)
uses commandio;

const
      cght=',/r/[//e/ jb/r/]//e/ jb/r/~//e/ jb/r/@//a/ jb/r/`//';
      cght2='/u/ jb/r/^//o/ jb/r/{/// jb/r/}/// jb/r/|//u/ jb"';

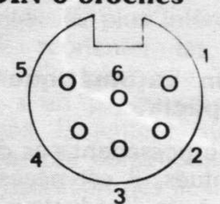
var
      fichier:string;

begin
  write(chr(7),' Nom du fichier : ');
  readln(fichier);
  chain(concat('I="E',fichier,'.TEXT',cght,cght2))
end.
```

Comme on peut le constater le POS recèle d'étonnantes possibilités. La manipulation des fichiers moniteurs et les modifications de ses entrées-sorties ne sont que deux des innombrables facultés de celui-ci. Le Pascal UCSD, quant à lui, possède également au niveau de sa syntaxe propre de nombreuses possibilités dont nous parlerons dans de prochains articles. ■

Brochage de la prise vidéo du TI-99

Fiche DIN 6 broches



1 - + 12 Volts
2 - Vidéo composite - standard NSTC
3 - R-Y
4 - B-Y
5 - Sortie audio
6 - Masse
R-Y et B-Y sont les signaux nécessaires à l'interface R.V.B. ■

Librairie Pascal

Lors de l'utilisation, en Pascal, de procédures ou de fonctions propres au TI-99 (manipulation des lutins et des couleurs, effets sonores, définitions de caractères, utilisation du synthétiseur de paroles), il est nécessaire de déclarer la librairie que l'on utilise. S'il s'agit de la librairie standard écrite par Texas Instruments (9 bibliothèques regroupées par "SYSTEM.LIBRARY") ne pas oublier, lors de la compilation, de déclarer le volume contenant ce fichier. En effet, "SYSTEM.LIBRARY" est pris par défaut, à condition qu'il se trouve en #4, lorsque le compilateur rencontre "USES BIBLIOTHEQUE". Si ce n'est pas le cas, il faut alors écrire :

(*$u NOMDUVOLUME:SYSTEM.LIBRARY *) ■

Un mode "test" sur le module Alpiner

Dans le numéro 3 de "99 Magazine", nous vous avions donoeé la méthode pour accéder au mode test de "Munch Man". Avec le module de jeu "Alpiner", il est possible de faire la même chose. Rappelons que la séquence : *#* (SHIFT-8, SHIFT-3 et SHIFT-8) permet l'entrée dans ce mode particulier.

Les particularités du Basic Etendu
Les Lutins (Sprites)

Julien Thomas

Le TI-99/4A est doté, pour la gestion de l'écran, d'un microprocesseur spécialisé (TMS 9918A) qui permet la création et l'animation de lutins. Pour le dictionnaire, les lutins sont des "petits génies espiègles, généralement bienveillants, parfois dangereux". Pour le TI-99, ce sont des objets auxquels nous pouvons donner certains paramètres, et qui apparaissent sur l'écran en superposition de l'affichage matriciel classique. Le module "Basic Etendu" comporte des sous-programme résidents (accessibles dès la mise sous tension) qui agissent directement sur les lutins.

Dans cet article, vous trouverez de nombreux petits programmes. Afin de faciliter leur chargement à partir de la cassette d'accompagnement, nous avons décidé de les regrouper dans un unique programme. Pour faire "tourner" ces exemples, vous devrez donc utiliser la commande "RUN numéro de ligne". Par exemple, pour CALL MAGNIFY, vous devrez taper : RUN 920. Pour sortir des programmes, il faudra utiliser "CLEAR" (FCTN-4).

CALL SPRITE

CALL SPRITE est l'instruction de base du système. Elle permet de créer un ou plusieurs lutins en leurs affectant une forme, une couleur, une position, etc... Sa syntaxe est :

- CALL SPRITE(#numéro du lutin,code de caractère,couleur,numéro de ligne,numéro de colonne[,vitesse verticale,vitesse horizontale]],autres lutins])

Les paramètres placés entre crochets sont facultatifs.

Le TI-99/4A peut générer jusqu'à 32 lutins, mais seulement 28 sont autorisés en Basic Etendu. Le numéro du lutin doit donc être un entier compris entre 1 et 28 inclus.

Le code de caractère doit être un entier compris entre 32 et 143, et correspond à un caractère qu'il est possible de redéfinir avec l'instruction CALL CHAR, tout comme en Basic TI.

Les couleurs sont les mêmes qu'en Basic TI (de 1 pour incolore à 16 pour blanc). Seule la couleur du lutin (points allumés) est modifiable, la couleur de fond (points éteints) étant toujours 1 (incolore).

Les numéros de ligne et de colonne déterminent la position du lutin sur l'écran lors de sa création, et correspondent à l'endroit où le coin supérieur gauche du lutins se trouvera.

Exemple :

```
100 ! *******************
110 ! *  CALL  SPRITE  *
120 ! *Un seul lutin fixe*
130 ! *******************
140 !
150 CALL CLEAR
160 CALL SPRITE(#1,65,7,94,124)
170 GOTO 170
```

Le moins que l'on puisse dire est que cet exemple n'est pas très impressionnant. Cependant, il démontre qu'il est possible de placer un lutin à n'importe quel endroit. Christian Monfort, dans son programme "Horloge", a très astucieusement utilisé cette possibilité. Puisque nous avons utilisé le code 65 pour créer notre lutin, il a la forme d'un "A".

Autre exemple :

```
180 ! *******************
190 ! *   CALL  SPRITE   *
200 ! * Un seul lutin en *
210 ! *    mouvement     *
220 ! *******************
230 !
240 CALL CLEAR
250 CALL SPRITE(#1,65,7,94,124,1
0,-10)
260 GOTO 260
```

Cette fois, le lutin se déplace et, ce qui est remarquable, sans qu'un programme ne soit obligé d'intervenir. Les vitesses de ligne et de colonne doivent être comprises entre −128 et 127. La valeur 0 donne un état fixe. Pour la vitesse de colonne, une valeur négative provoque un déplacement de droite à gauche, alors qu'une valeur positive provoque un déplacement de gauche à droite. Pour la vitesse de ligne, une valeur négative donne un déplacement du bas vers le haut, alors qu'une vitesse positive provoque un déplacement de haut en bas. En jouant sur les vitesses de ligne et de colonne, il est possible d'obtenir des déplacements dans n'importe quelle direction.

```
270 ! *******************
280 ! *   CALL  SPRITE   *
290 ! *  Deux lutins en  *
300 ! *    mouvement     *
310 ! *******************
320 !
330 CALL CLEAR
340 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
350 GOTO 350
```

Cette fois, nous avons deux lutins se déplaçant dans des directions opposées. En fait, seule la place disponible dans une ligne de programme limite le nombre de lutins qu'il est possible de créer dans une seule instruction CALL SPRITE.

CALL COLOR

Lorsque l'on a fait apparaître un lutin avec CALL SPRITE, il peut être intéressant de changer sa couleur. C'est ce que fait CALL COLOR, qui s'utilise de la manière suivante :

- CALL COLOR(#numéro de lutin,couleur[,autre lutin])

Cette instruction n'est pas la même que le CALL COLOR pour modifier la couleur d'un groupe de caractères. Si vous essayez de mélanger les caractères et les lutins dans cette instruction, vous obtiendrez un splendide message d'erreur. Dans l'exemple suivant, nos deux lutins changent continuellement de couleur de façon aléatoire.

```
360 ! *******************
370 ! *    CALL COLOR    *
380 ! *******************
390 !
400 CALL CLEAR
410 CALL SCREEN(1)
420 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
430 RANDOMIZE :: CALL COLOR(#1,I
NT(RND*14)+3,#2,INT(RND*14)+3)::
 GOTO 430
```

CALL CHAR

Jusqu'à présent, nous nous étions contentés, pour nos deux lutins, de caractères prédéfinis. L'instruction CALL CHAR autorise un "remodelage" des caractères et, si elle s'utilise de la même façon qu'en Basic TI, elle comporte néanmoins quelques différences. En Basic TI, un CALL CHAR ne peut redéfinir qu'un seul caractère. Ici, on peut redéfinir plusieurs caractères simultanément et une chaîne (codage hexadécimal de la forme) peut contenir jusqu'à 64 éléments :

- si la chaîne contient moins de 16 éléments, un seul caractère est redéfini;
- si elle contient entre 17 et 32 éléments, deux caractères sont redéfinis;
- si elle contient entre 33 et 48 éléments, trois caractères;
- si elle contient entre 49 et 64 éléments, quatre caractères.

Dans tous les cas, les éléments manquants sont remplacés par des zéros. Si une chaîne contient plus de 64 éléments, le surplus est ignoré. Lorsque plusieurs caractères sont définis simultanément, l'ordre suivant est respecté :

X	X+2
X+1	X+3

Le petit segment de programme suivant tranforme les lutins des exemples précédents en un rectangle et un losange :

```
440 ! *******************
450 ! *    CALL  CHAR    *
460 ! *******************
470 !
480 CALL CLEAR
490 CALL SCREEN(1)
500 CALL CHAR(65,"FF8181818181818l
FF183C66C3C3663C18")
510 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
520 RANDOMIZE :: CALL COLOR(#1,I
NT(RND*14)+3,#2,INT(RND*14)+3)::
 GOTO 520
```

CALL MOTION

Cette instruction permet de modifier le déplacement d'un ou de plusieurs lutins, ou de mettre en mouvement un ou plusieurs lutins immobiles. La syntaxe est très simple :

- CALL MOTION(#numéro de lutin,vitesse de ligne,vitesse de colonne[,autre lutin])

Dans le premier exemple, les vitesses sont choisies aléatoirement. Notons au passage que INT(RND*20)*SGN(RND−.5) retourne une valeur comprise entre −19 et 19 puisque SGN(RND−.5) retourne −1, 0 ou 1.

```
530 ! *******************
540 ! *    CALL  MOTION   *
550 ! * premier  exemple *
560 ! *******************
570 !
580 CALL CLEAR
590 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
600 RANDOMIZE :: CALL MOTION(#1,
INT(RND*20)*SGN(RND-.5),INT(RND*
20)*SGN(RND-.5),
#2,INT(RND*20)*SGN(RND-.5),INT(R
ND*20)*SGN(RND-.5)):: GOTO 600
```

Le deuxième exemple autorise un déplacement contrôlé. Vous devez appuyer sur les touches fléchées pour obtenir le déplacement du lutin.

```
610 ! *******************
620 ! *    CALL  MOTION   *
630 ! * deuxieme exemple *
640 ! *******************
650 !
660 CALL CLEAR
670 CALL SCREEN(1)
680 CALL CHAR(32,"80808080808080
FFFF818181818181FF")
690 CALL COLOR(1,4,8)
700 CALL SPRITE(#1,33,7,94,124)
710 CALL KEY(0,K,S):: IF S=0 THE
N X=0 :: Y=0 :: GOTO 760
720 IF K=69 THEN X=-10 :: Y=0 ::
 GOTO 760
730 IF K=83 THEN X=0 :: Y=-10 ::
 GOTO 760
740 IF K=68 THEN X=0 :: Y=10 ::
GOTO 760
750 IF K=88 THEN X=10 :: Y=0 ELS
E 710
760 CALL MOTION(#1,X,Y):: GOTO 7
10
```

CALL POSITION

Cette instruction vous permet de savoir où se trouvent les lutins. Les positions verticales et horizontales sont retournées dans des variables. CALL POSITION s'utilise ainsi :

- CALL POSITION(#numéro du lutin,variable ligne,variable colonne[,autre lutin])

Le petit exemple placé ci-dessous met deux lutins en mouvement et reporte en permanence leurs positions. USING, associé à DISPLAY autorise un affichage "formaté" :

```
770 ! *******************
780 ! *  CALL  POSITION  *
790 ! *******************
800 !
810 CALL CLEAR
820 DISPLAY AT(21,2):"1er LUTIN
     2eme LUTIN" :: DISPLAY AT(
23,3):"Ligne:
       Ligne:" :: DISPLAY AT(24,
1):"Colonne:        Colonne:"
830 DISPLAY AT(22,2):"---------
     ---------"
840 CALL SPRITE(#1,65,7,94,124,1
0,-10,#2,66,13,94,124,10,10)
850 RANDOMIZE :: CALL MOTION(#1,
INT(RND*20)*SGN(RND-.5),INT(RND*
20)*SGN(RND-.5),
#2,INT(RND*20)-SGN(RND-.5),INT(R
ND*20)*SGN(RND-.5))
860 CALL POSITION(#1,X1,Y1,#2,X2
,Y2):: DISPLAY AT(23,10)SIZE(3):
USING "###":X1 :
: DISPLAY AT(24,10)SIZE(3):USING
 "###":Y1
870 DISPLAY AT(23,26):USING "###
":X2 :: DISPLAY AT(24,26):USING
"###":Y2 :: GOTO
 850
```

CALL MAGNIFY

C'est l'instruction qui possède la syntaxe la plus simple :

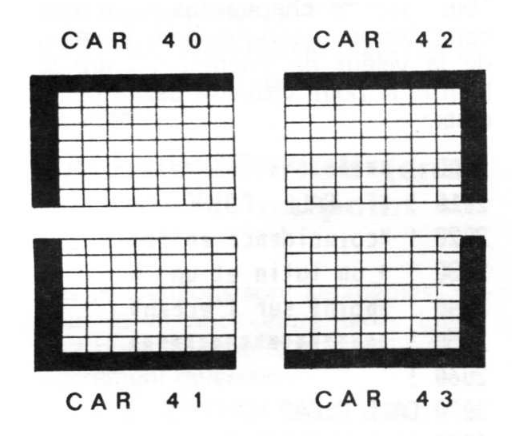
- CALL MAGNIFY(valeur)

La valeur doit être un entier de 1 à 4. Examinons les effets de cette instruction :

1 - est la valeur par défaut, donc celle que nous utilisons depuis le début de cet article. Le lutin utilise un seul code de caractère et occupe la même surface qu'un caractère standard;
2 - le lutin utilise toujours un seul code de caractère mais, cette fois, chaque point définissant le lutin occupe quatre points d'écran (pixels), ce qui fait que l'on couvre maintenant une surface identique à celle nécessaire à la représentation de quatre caractères standards;
3 - cette fois, comme pour CALL MAGNIFY(1), les points du lutin occupent un point sur l'écran. Par contre, quatre codes de caractères sont utilisés. Le premier caractère est celui qui représente le quart supérieur gauche du lutin et son code doit être divisible par quatre. Si tel n'est pas le cas, le premier caractère utilisé sera le plus proche à posséder un code divisible par quatre, en al-

lant vers les nombres les plus forts;
4 - il s'agit là de la combinaison des valeurs 2 et 3. En effet, les lutins utilisent quatre codes de caractères et chacun de leurs points occupe quatre points sur l'écran. Chaque lutin utilise une surface équivalente à celle nécessaire à la représentation de seize caractères standards.

Le petit programme suivant visualise l'action de CALL MAGNIFY. Les caractères utilisés sont redéfinis ainsi :



Les touches 1, 2, 3 et 4 retournent un "CALL MAGNIFY" correspondant :

```
880 ! ********************
890 ! *   CALL MAGNIFY    *
900 ! ********************
910 !
920 CALL CLEAR
930 CALL CHAR(40,"FF808080808080
80808080808080FFFF010101010101
0101010101010101
FF")
940 CALL SPRITE(#1,40,7,94,124)
950 CALL KEY(0,K,S):: IF (K<49)O
R(K>52)THEN 950 ELSE CALL MAGNIF
Y(K-48):: GOTO 9
50
```

CALL PATTERN

Cette instruction est utilisée pour modifier l'apparence d'un ou de plusieurs lutins. Ceci est très utile pour animer des objets ou des personnages. La modification du lutin est effectuée en lui affectant un code de caractère différent à chaque utilisation de CALL PATTERN. Syntaxe de l'instruction :

- CALL PATTERN(#numéro du lutin,code de caractère[,autre lutin])

Le premier exemple proposer pour illustrer cette instruction anime un objet volant parfaitement identifié, puisqu'il s'agit de la soucoupe du jeu "TI-Invaders" :

```
960 ! ********************
970 ! *   CALL PATTERN   *
980 ! * Premier exemple *
990 ! ********************
1000 !
1010 CALL CLEAR
1020 CALL SCREEN(1)
1030 CALL MAGNIFY(3)
1040 DATA 01033FFF77FF3F07,0,80C
0FCFF77FFFCE,0,01033FFFBBFF3F07,
0,80C0FCFFBBFFFC
E,0
1050 DATA 01033FFFDDFF3F07,0,80C
0FCFFDDFFFCE,0,01033FFFEEFF3F07,
0,80C0FCFFEEFFFC
E,0
1060 FOR BD=40 TO 55 :: READ C$
:: CALL CHAR(BD,C$):: NEXT BD
1070 CALL SPRITE(#1,40,11,90,10,
0,9)
1080 FOR BP=40 TO 52 STEP 4 :: C
ALL PATTERN(#1,BP):: FOR D=0 TO
5 :: NEXT D :: N
EXT BP :: GOTO 1080
```

Le second exemple représente un petit personnage en marche et, s'il ressemble à "E.T.", ce n'est cette fois qu'un pur hasard :

```
1090 ! ******************
1100 ! *  CALL  PATTERN  *
1110 ! *Deuxieme exemple*
1120 ! ******************
1130 !
1140 CALL CLEAR
1150 CALL SCREEN(13)
1160 CALL CHAR(40,"1C1C1038385C5
C9A9A181810101030180000000000000
0000")
1170 CALL CHAR(44,"1C1C103838385
C5C5C181412242426300000000000000
0000")
1180 CALL CHAR(48,"1C1C103838383
8383818182824446406000000000000
0000")
1190 CALL CHAR(52,"1C1C103838385
C5C5C18181828482800C000000000000
0000")
1200 CALL COLOR(1,5,5):: CALL MA
GNIFY(3):: CALL SPRITE(#1,40,16,
177,1):: X=1
1210 X=1
1220 FOR B=40 TO 52 STEP 4 :: CA
LL PATTERN(#1,B)
1230 X=X+1 :: IF X=256 THEN 1210
1240 CALL LOCATE(#1,177,X)
1250 FOR D=0 TO 1 :: NEXT D
1260 NEXT B
1270 GOTO 1220
```

CALL LOCATE

Il est possible, grâce à cette instruction, de placer un lutin à n'importe quel endroit sur l'écran. Une instruction CALL LOCATE est composée comme suit :

- CALL LOCATE(#numéro du lutin,numéro de ligne,numéro de colonne[,autre lutin])

Dans le programme suivant, un lutin est créé au milieu de l'écran et, au bout de quelques secondes, il se met à se déplacer suivant un parcours elliptique :

```
1280 ! ******************
1290 ! *   CALL LOCATE   *
1300 ! ******************
1310 !
1320 CALL CLEAR
1330 CALL CHAR(32,"000000FFFF000
0003C7EFFFFFFFF7E3C")
1340 CALL COLOR(1,5,8)
1350 CALL SCREEN(1)
1360 B1=PI/180 :: B2=B1*360 :: B
3=B1*3
1370 CALL SPRITE(#1,33,7,94,124)
:: FOR BD=0 TO 500 :: NEXT BD
1380 FOR B=B1 TO B2 STEP B3 :: C
ALL LOCATE(#1,98+40*SIN(B),128+9
0*COS(B)):: NEXT
 B :: GOTO 1380
```

CALL COINC(ALL,variable)

Ceci autorise la détection d'une éventuelle coïncidence entre deux lutins. Dans le cas de l'utilisation du paramètre "ALL", le chevauchement de n'importe quel lutin avec n'importe quelle autre lutin retoune la valeur −1 dans la variable utilisée. Si aucune coïncidence n'est détectée, la variable contient 0.

Le premier exemple met en évidence que, lorsqu'on utilise "ALL", pour qu'une coïncidence soit détectée, il faut une superposition physique. Les coordonnées des lutins sont les mêmes pour les quatre lutins mais, comme les lutins ne se chevauchent pas, la variable "C" contient 0.

```
1390 ! ******************
1400 ! *   CALL  COINC   *
1410 ! * (ALL, variable) *
1420 ! * premier exemple *
1430 ! ******************
1440 !
1450 CALL CLEAR
1460 CALL COLOR(1,16,8)
1470 CALL SCREEN(4)
1480 CALL CHAR(32,"1824428181422
418")
1490 CALL MAGNIFY(4)
```

```
1500 DATA FF8080808080808,808080
80808080FF,FF0101010101010101,0101
0101010101FF
1510 DATA 007F40404040404,404040
4040407F,00FE0202020202,020202
020202FE
1520 DATA 00003F20202020202,202020
20203F,0000FC0404040404,04040404
04FC
1530 DATA 0000001F1010101,101010
101F,000000F808080808,08080808F8
1540 RESTORE 1500
1550 FOR BD=96 TO 111 :: READ C$
 :: CALL CHAR(BD,C$):: NEXT BD
1560 FOR BL=1 TO 4 :: CALL SPRIT
E(#BL,92+BL*4,BL+11,81,113):: NE
XT BL
1570 CALL COINC(ALL,C):: IF C TH
EN DISPLAY AT(23,1):"COINCIDENCE
" ELSE DISPLAY A
T(23,1):"PAS DE COINCIDENCE"
1580 GOTO 1570
```

Dans le deuxième exemple, bien que les coordonnées de deux lutins soient différentes, la variable "C" contient −1 puisque le second lutin chevauche légèrement le premier.

```
1590 ! ******************
1600 ! *   CALL  COINC   *
1610 ! * (ALL, variable) *
1620 ! *deuxieme  exemple*
1630 ! ******************
1640 !
1650 CALL CLEAR
1660 CALL SCREEN(4)
1670 CALL COLOR(1,16,8)
1680 CALL MAGNIFY(4)
1690 CALL CHAR(32,"1824428181422
418",96,RPT$("F",64))
1700 CALL SPRITE(#1,96,9,64,96,#
2,96,5,95,127)
1710 CALL COINC(ALL,C):: IF C TH
EN DISPLAY AT(23,1):"COINCIDENCE
" ELSE DISPLAY A
T(23,1):"PAS DE COINCIDENCE"
1720 GOTO 1710
```

Le troisième exemple met en mouvement, de manière aléatoire, 28 lutins. En suite, à chaque fois que deux ou plusieurs d'entre eux se rencontrent, le programme émet un petit "bip" sonore.

```
1730 ! ******************
1740 ! *   CALL  COINC   *
1750 ! * (ALL, variable) *
1760 ! *troisieme exemple*
1770 ! ******************
1780 !
1790 CALL CLEAR
1800 CALL SCREEN(1)
1810 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
1820 DATA 3,4,5,6,7,8,9,10,11,12
,13,14,15,16,3,4,5,6,7,8,9,10,11
,12,13,14,15,16
1830 RESTORE 1820
1840 FOR B=1 TO 28 :: READ C ::
RANDOMIZE :: CALL SPRITE(#B,40,C
,100,100,INT(RND
*20)*SGN(RND-.5),INT(RND*20)*SGN
(RND-.5)):: NEXT B
1850 CALL COINC(ALL,V):: IF V TH
EN CALL SOUND(1,1000,0)
1860 GOTO 1850
```

CALL COINC (coïncidence entre deux lutins)

La syntaxe est différente de celle utilisée au paragraphe précédent :

- CALL COINC(#numéro de lutin,#numéro de lutin,tolérance,variable)

La tolérance indique au programme à partir de quelle distance entre les deux lutins il devra considérer qu'il y a coïncidence. Concrètement, si vous donnez 20 comme valeur de tolérance, et que les lutins se trouvent à moins de 20 points l'un de l'autre, la variable utilisée dans l'instruction contiendra −1, sinon, est contiendra 0. Cette fois, la détection se fait non plus de manière physique comme avec "ALL", mais de manière logique : le calcul se fait par rapport aux coordonnées des lutins. De façon générale, pour que les coïncidences soient reconnues, la valeur de tolérance doit être directement proportionnelle à la vitesse des lutins.

```
1870 ! ******************
1880 ! *   CALL  COINC   *
1890 ! *coincidence entre*
1900 ! *   deux lutins   *
1910 ! ******************
1920 !
1930 CALL CLEAR
1940 CALL MAGNIFY(2)
1950 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
1960 CALL SPRITE(#1,40,9,100,100
,#2,40,5,100,100)
1970 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
,#2,INT(RND*20)*SGN(RND-.5),INT(
RND*20)*SGN(RND-.5))
1980 CALL COINC(#1,#2,20,C):: IF
 C THEN CALL SOUND(1,1000,0)
1990 GOTO 1970
```

CALL COINC (coïncidence entre un lutin et un point sur l'écran)

Cette instruction agit de la même manière que celle rencontrée dans le paragraphe précédent, le deuxième lutin est simplement remplacé par les coordonnées d'un point :

- CALL COINC (#numéro du lutin,ligne,colonne,tolérance,variable)

Le programme suivant émet un "bip" sonore chaque fois qu'il y a coïncidence théorique (en fonction de la valeur de tolérance) entre le lutin et le point visualisé par la petite croix :

```
2000 ! ******************
2010 ! *   CALL  COINC   *
2020 ! *coincidence entre*
2030 ! * un lutin et un  *
2040 ! *point sur l'ecran*
2050 ! ******************
2060 !
2070 CALL CLEAR
2080 CALL MAGNIFY(2)
2090 CALL CHAR(40,"FF81818181818
1FF101010FE10101")
2100 CALL COLOR(2,7,8)
2110 CALL HCHAR(12,16,41)
2120 CALL SPRITE(#1,40,5,100,100
)
2130 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
)
2140 CALL COINC(#1,92,124,20,C):
: IF C THEN CALL SOUND(1,1000,0)
2150 GOTO 2130
```

CALL DISTANCE (entre deux lutins)

Cette instruction, dont voici la syntaxe :

- CALL DISTANCE(#numéro de lutin,#numéro de lutin,variable)

retourne le carré de la distance entre les coins supérieurs des deux lutins. Les nombres de colonnes et de lignes séparant les lutins sont élevés au carré et additionnés. Si le produit de ce calcul est supérieur à 32767, la valeur 32767 est retournée. La distance réelle entre les lutins est la racine carrée de la valeur retournée.

```
2160 ! ******************
2170 ! * CALL DISTANCE  *
2180 ! * distance  entre *
2190 ! *   deux lutins   *
2200 ! ******************
```

```
2210 !
2220 CALL CLEAR
2230 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
2240 DISPLAY AT(24,1):"DISTANCE:
"
2250 CALL SPRITE(#1,40,9,100,100
,#2,40,5,100,100)
2260 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
,#2,INT(RND*20)*SGN(RND-.5),INT(
RND*20)*SGN(RND-.5))
2270 CALL DISTANCE(#1,#2,V):: DI
SPLAY AT(24,11):INT(SQR(V)):: GO
TO 2260
```

CALL DISTANCE (entre un lutin et un point sur l'écran)

Les conventions sont les mêmes que pour la distance entre deux lutins, sauf pour la syntaxe qui est bien sûr légèrement différente :

- CALL DISTANCE(#numéro de lutin,ligne,colonne,variable)

```
2280 ! ******************
2290 ! *  CALL DISTANCE  *
2300 ! * distance  entre *
2310 ! * un lutin et un  *
2320 ! *point sur l'ecran*
2330 ! ******************
2340 !
2350 CALL CLEAR
2360 CALL COLOR(2,7,B)
2370 CALL HCHAR(12,16,41)
2380 CALL CHAR(40,"FFFFFFFFFFFF
FFF101010FE101010")
2390 DISPLAY AT(24,1):"DISTANCE:
"
2400 CALL SPRITE(#1,40,9,100,100
)
2410 RANDOMIZE :: CALL MOTION(#1
,INT(RND*20)*SGN(RND-.5),INT(RND
*20)*SGN(RND-.5)
)
2420 CALL DISTANCE(#1,92,124,V):
: DISPLAY AT(24,11):INT(SQR(V)):
: GOTO 2410
```

CALL DELSPRITE (tous les lutins)

Cette instruction élimine tous les lutins présents sur l'écran :

- CALL DELSPRITE(ALL)

```
2430 ! ******************
2440 ! * CALL  DELSPRITE *
2450 ! *      (ALL)      *
2460 ! ******************
2470 !
2480 CALL CLEAR
2490 CALL SCREEN(1)
2500 FOR B=1 TO 28 :: RANDOMIZE
2510 CALL SPRITE(#B,B+64,INT(RND
*14)+3,INT(RND*192)+1,INT(RND*25
6)+1,INT(RND*20)
*SGN(RND-.5),INT(RND*20)*SGN(RND
-.5)):: NEXT B
2520 FOR D=0 TO 1000 :: NEXT D
2530 CALL DELSPRITE(ALL):: FOR D
=0 TO 500 :: NEXT D :: GOTO 2500
```

CALL DELSPRITE (un ou plusieurs lutins)

Cette version autorise une utilisation plus sélective que CALL DELSPRITE(ALL) :

- CALL DELPRITE(#numéro de lutin[,autre lutin])

L'exemple suivant crée 28 lutins qu'il efface ensuite un par un, en partant de celui dont le numéro est le plus élevé :

```
2540 ! ******************
2550 ! * CALL  DELSPRITE *
2560 ! *    un lutin     *
2570 ! ******************
2580 !
2590 CALL CLEAR
2600 CALL SCREEN(1)
2610 CALL CHAR(40,"FFFFFFFFFFFF
FFF")
2620 FOR B=1 TO 28 :: RANDOMIZE
2630 CALL SPRITE(#B,40,INT(RND*1
4)+3,INT(RND*192)+1,INT(RND*256)
+1,INT(RND*20)*S
GN(RND-.5),INT(RND*20)*SGN(RND-.
5)):: NEXT B
2640 FOR D=0 TO 1000 :: NEXT D
2650 FOR B=28 TO 1 STEP -1 :: CA
LL DELSPRITE(#B):: FOR D=0 TO 20
0 :: NEXT D :: N
EXT B :: FOR D=0 TO 500 :: NEXT
D :: GOTO 2620
```

Superposition des lutins

Les lutins, ce n'est un secret pour personne, se superposent à l'écran graphique classique. En plus, si deux lutins occupent une zone d'écran commune, celui qui possède le numéro le moins élevé recouvrira l'autre :

```
2660 ! ******************
2670 ! *  SUPERPOSITION  *
2680 ! *   DES LUTINS    *
2690 ! ******************
2700 !
2710 CALL CLEAR
2720 CALL SCREEN(1)
2730 CALL MAGNIFY(2)
2740 CALL CHAR(32,"1824428181422
418FFFFFFFFFFFFFF")
2750 CALL COLOR(1,16,8)
2760 L=20 :: C=24
2770 DATA 2,3,5,7,4,6,9,13,10,11
,14,12,15,16,2,3,5,7,4,6,9,13,10
,11,14,12,15,16
2780 RESTORE 2770
2790 FOR B=1 TO 28 :: READ CO ::
 CALL SPRITE(#B,33,CO,L,C):: L=L
+5 :: C=C+7 :: N
EXT B
2800 GOTO 2800
```

Disparition partielle de certains lutins

Lorsque plus de 4 lutins sont situés sur la même ligne d'écran, celui ou ceux qui ont les numéros les plus élevés disparaissent :

```
2810 ! ******************
2820 ! *   DISPARITION   *
2830 ! *  PARTIELLE DE   *
2840 ! * CERTAINS LUTINS *
2850 ! ******************
2860 !
2870 CALL CLEAR
2880 CALL SCREEN(1)
2890 CALL MAGNIFY(2)
2900 CALL CHAR(32,"1824428181422
418FFFFFFFFFFFFFF")
2910 CALL COLOR(1,16,8)
2920 L=20 :: C=24
2930 DATA 2,3,5,7,4,6,9,13,10,11
,14,12,15,16,2,3,5,7,4,6,9,13,10
,11,14,12,15,16
2940 RESTORE 2930
2950 FOR B=1 TO 28 :: READ CO ::
 CALL SPRITE(#B,33,CO,L,C):: L=L
+5 :: C=C+7 :: N
EXT B
2960 FOR D=0 TO 1000 :: NEXT D :
: FOR B=1 TO 28 :: RANDOMIZE ::
CALL MOTION(#B,I
NT(RND*20)*SGN(RND-.5),INT(RND*2
0)*SGN(RND-.5)):: NEXT B
2970 GOTO 2970
```

Détection de la présence de cinq lutins ou plus, sur la même ligne

Aucune instruction du Basic Etendu ne permet de faire ce genre de chose. L'instruction PEEK, qui autorise la lecture du contenu d'une

adresse mémoire, nous permet de combler cette lacune.
L'adresse >837B (−31877 en décimal) contient, entre autre, le numéro du cinquième lutin s'il y a plus de quatre lutins sur la même ligne. Ce numéro est localisé dans les bits 3 à 7 (si vous désirez obtenir plus de renseignements sur cette adresse, reportez-vous à l'article de Denise Amrouche : Les utilisations de la Mini-mémoire). CALL PEEK et les opérateurs logiques AND et NOT remplacent facilement cette instruction oubliée.
Le programme suivant efface un lutin lorsque cinq lutins ou plus sont situés sur la même ligne. Le numéro du lutin effacé le plus récemment est affiché au bas de l'écran :

```
2980 ! ******************
2990 ! * DETECTION DE LA *
3000 ! *   PRESENCE DE   *
3010 ! *   CINQ LUTINS   *
3020 ! * OU PLUS, SUR LA *
3030 ! *   MEME  LIGNE   *
3040 ! ******************
3050 !
3060 CALL CLEAR
3070 CALL SCREEN(1)
3080 CALL CHAR(32,"1824428181422
418")
3090 CALL COLOR(1,5,1,3,8,1,4,8,
1)
3100 CALL MAGNIFY(2)
3110 FOR B=1 TO 28 :: RANDOMIZE
:: CALL SPRITE(#B,B+64,INT(RND*1
4)+3,INT(RND*192
)+1,INT(RND*256)+1,INT(RND*20)*S
GN(RND-.5),INT(RND*20)*SGN(RND-.
5)):: NEXT B
3120 FOR D=0 TO 1000 :: NEXT D
3130 CALL PEEK(-31877,A):: IF NO
T A AND 64 THEN 3130
3140 L=(A AND 31)+1 :: DISPLAY A
T(24,1):L :: CALL DELSPRITE(#L):
: CALL SOUND(1,1
000,0):: GOTO 3130
```

Conclusion

Nous disposons, avec les lutins, d'un système graphique puissant. Cependant, il ne faut pas se faire d'illusions; le Basic Etendu est beaucoup trop lent pour permettre la création de jeux d'action très rapides. Pour cela, il faudra passer à l'Assembleur. Les lutins sont malgré tout très pratiques pour améliorer la présentation des programmes, réaliser des simulations ou même des jeux, à condition que les déplacements soient programmés définitivement ou qu'ils soient suffisamment lents pour que les instructions de détection puissent agir. ■

Chargeur automatique en Basic Etendu

Eric Thomas

Ce petit programme, s'il est placé dans le premier lecteur de disquettes sous le nom "LOAD", permet d'exécuter directement n'importe quel programme dès que l'on accède au Basic Etendu. En effet, le programme "LOAD", s'il existe, est automatiquement exécuté. D'autre part, comme le Basic Etendu permet d'enchaîner plusieurs programmes au moyen de l'instruction "RUN", rien ne nous empêche de créer sur chaque disquette un programme affichant automatiquement son contenu (par un menu), et permettant de lancer directement l'exécution du programme choisi.

Pour que ce "chargeur" soit général et puisse fonctionner avec n'importe quelle disquette, il doit lire le fichier "DSK1." qui contient notamment les noms et types de tous les fichiers de la disquette.
Seuls sont affichés les fichiers dont l'entier déterminant le type vaut 5 ou −5, ce qui correspond à un fichier de type programme (la valeur est négative lorsque le fichier est protégé). Il ne resterait alors plus qu'à écrire "RUN A$", par exemple, où "A$" serait le nom du programme sélectionné. Mais c'est là que les choses se compliquent. Il est en effet impossible d'écrire RUN suivi d'une chaîne contenant le nom du programme à exécuter. Le TI ne peut comprendre un RUN que lorsque le nom du programme est écrit EXPLICITEMENT.

Pour garder un caractère général et fonctionner sans modification ou mise à jour d'une disquette à une autre, le programme "poke" directement le nom du programme à exécuter. Il modifie la dernière ligne du programme (RUN "DSK1.??????????) en écrivant explicitement le nom du programme choisi à la place des points d'interrogation.

Pour transformer correctement la dernière ligne, il est nécessaire de :

1) "poker" le numéro du lecteur contenant le programme (c'est la réponse à la question QUEL DISQUE ?);
2) changer l'octet indiquant au système la longueur de la chaîne située après le "RUN" (5 + la longueur du nom du programme à exécuter);
3) remplacer les points d'interrogation par les lettres composant le nom du programme;
4) "poker" un zéro derrière la dernière lettre pour marquer la fin de la ligne (les points d'interrogation résiduels sont ignorés).

ATTENTION : si vous modifiez ce programme (suppression des remarques, présentation différente du menu), il faut recalculer l'adresse du point de RUN "DSK1.??????????" (au moyen de PEEK), et l'indiquer au programme en modifiant la valeur de "M" qui est donnée ligne 200. ■

```
100 ! *********************
110 ! *      Chargeur     *
120 ! *   Basic Etendu    *
130 ! *  a usage general  *
140 ! *********************
150 ! *      Copyright    *
160 ! * Gerard Santraille *
170 ! * et  "99 Magazine" *
180 ! *********************
190 !
200 CALL INIT :: M=-345 :: OPTION BASE 1
:: DIM P$(20)
210 CALL SCREEN(15):: DISPLAY AT(12,8)ERA
SE ALL:"Quel disque : 1" :: ACCEPT AT(1
2,22)BEEP SIZE(-1)VALIDATE("123"):D$
220 OPEN #1:"DSK"&D$&".",INPUT ,RELATIVE,
INTERNAL :: INPUT #1:N$,A,A,A :: DISPLA
Y AT(1,1)ERASE ALL:N$;"  ";A;"secteurs li
bres"
230 M$="## ---> "&RPT$("#",10)
240 FOR I=1 TO 20
250 INPUT #1:N$,A,B,B :: IF N$="" THEN 29
0
260 IF ABS(A)=5 THEN DISPLAY AT(I+2,5):US
ING M$:I,N$ :: P$(I)=N$ ELSE 250
270 NEXT I
280 DISPLAY AT(I+2,5):USING M$:I,"CONTINU
ER"
290 DISPLAY AT(24,13):"VOTRE CHOIX :" ::
ACCEPT AT(24,27)BEEP SIZE(-2):N :: IF I
=N THEN CALL HCHAR(3,1,32,704):: GOTO 240
300 CLOSE #1 :: CALL LOAD(M-1,ASC(D$))
310 FOR P=1 TO LEN(P$(N)):: CALL LOAD(M+P
,ASC(SEG$(P$(N),P,1))):: NEXT P :: CALL
 LOAD(M-5,4+P):: CALL LOAD(M+P,0)
320 RUN "DSK1.??????????"
```

Le Solitaire

Olivier Colin

Le principe de ce jeu, écrit en **Basic TI**, est exactement celui du "solitaire". Mais il comporte des avantages : vous n'aurez pas, en jouant, à placer et à déplacer vous-même les pions, le TI étant à votre disposition pour cela. Votre seul travail consiste à entrer le numéro de la ligne et de la colonne de départ, puis à faire de même pour la case d'arrivée.

Le programme commence par l'élaboration des formes (pions, tableau et titre), aux lignes 240 à 380. Ensuite, il se branche en ligne 2360 à l'aide d'un GOSUB pour faire apparaître le titre, qui lui-même est suivi de la règle du jeu (lignes 2540 à 2640).

Les lignes 410 à 430 initialisent les variables. Puis le quadrillage se construit (lignes 480 à 640) et apparaît grâce à la routine de "transparence" (lignes 1970 à 2000), et à la routine "réapparition des couleurs" (lignes 2040 à 2070). Ce système a déjà été utilisé pour l'apparition du titre et de la règle du jeu. Les lignes 710 à 770 impriment les numéros des lignes et des colonnes (1 à 8). Le jeu commence à la ligne 810.

L'ordinateur affiche "DEPART", à l'aide d'un sous-programme expliqué plus bas, et émet un petit bip. Celui-ci vous annonce que vous pouvez entrer l'abscisse des coordonnées de départ. Dès que vous avez entré le premier chiffre à l'aide d'un CALL KEY (ce n'est donc pas la peine d'appuyer sur la touche "ENTER"), un second bip annonce que vous pouvez entrer l'ordonnée des coordonnées de départ. Après, il faut procéder de même pour l'entrée des coordonnées de la case d'arrivée du pion.

Les lignes 1300 à 1420 servent au déplacement du pion. Un caractère vide se met à la place du pion à déplacer (coordonnées de départ) qui s'affiche à la case d'arrivée. Le pion sauté est remplacé par un blanc. C'est à la ligne 1420 que le programme se branche à la ligne 2110 pour augmenter le compteur de 10 points.

Mais avant que le pion ne se déplace, il faut procéder à plusieurs opérations :

1 - vérifier que le caractère qui saute est un pion. La réponse à cette question se trouve dans les lignes 1480 à 1500 et utilise l'instruction CALL GCHAR;
2 - vérifier que le caractère d'arrivée est vide (lignes 1560 à 1640);
3 - vérifier que le caractère sauté est un pion (lignes 1700 à 1840).

Ces vérifications sont effectuées par les sous-programmes appelés au moment des entrées des coordonnées.
Le sous-programme situé aux lignes 1880 à 1930 permet d'afficher une suite de caractères (ce sous-programme correspond au "DISPLAY AT" du Basic Etendu). ■

```
100 REM ******************
110 REM *  LE  SOLITAIRE  *
120 REM ******************
130 REM *    BASIC  TI    *
140 REM ******************
150 REM *    COPYRIGHT    *
160 REM *  OLIVIER COLIN  *
170 REM * ET  99 Magazine *
180 REM ******************
190 REM******************
200 CALL CLEAR
210 REM **************
220 REM INITIALISATION
230 REM **************
240 CALL CHAR(97,"1010101010101010")
250 CALL CHAR(98,"00000000FF000000")
260 CALL CHAR(99,"10101010FF101010")
270 CALL CHAR(100,"000000001F101010")
280 CALL CHAR(101,"00000000FF101010")
290 CALL CHAR(102,"101010101F")
300 CALL CHAR(103,"10101010FF")
310 CALL CHAR(104,"101010101F101010")
320 CALL CHAR(105,"00000000F0101010")
330 CALL CHAR(106,"10101010F0")
340 CALL CHAR(107,"10101010F0101010")
350 CALL CHAR(108,"")
360 CALL CHAR(122,"00547C1010107CFE")
370 CALL CHAR(130,"FFFFFFFFFFFFFFFF")
380 CALL CHAR(131,"FFFCF0C0C0F0FCFF")
390 GOSUB 2360
400 CALL CLEAR
410 S=5
420 T=8
430 SCORE=0
440 GOSUB 1970
450 REM *******
460 REM TABLEAU
470 REM *******
480 PRINT TAB(20);"SCORE:";SCORE: : : :
490 PRINT TAB(8);"llldbebebilll"
500 PRINT TAB(8);"lllazazazalll"
510 PRINT TAB(8);"lllhbcbcbklll"
520 PRINT TAB(8);"lllazazazalll"
530 PRINT TAB(7);"dbebcbcbcbcbebi"
540 PRINT TAB(7);"azazazazazazaza"
550 PRINT TAB(7);"hbcbcbcbcbcbcbk"
560 PRINT TAB(7);"azazaza azazaza"
570 PRINT TAB(7);"hbcbcbcbcbcbcbk"
580 PRINT TAB(7);"azazazazazazaza"
590 PRINT TAB(7);"fbgbcbcbcbcbgbj"
600 PRINT TAB(8);"lllazazazalll"
610 PRINT TAB(8);"lllhbcbcbklll"
620 PRINT TAB(8);"lllazazazalll"
630 PRINT TAB(8);"lllfbgbgbjlll"
640 PRINT : : :
650 CALL SCREEN(11)
660 GOSUB 2040
670 CALL COLOR(12,14,1)
680 REM *****************
690 REM LIGNES ET COLONNES
700 REM *****************
710 FOR COMP=49 TO 55
720 S=S+2
730 T=T+2
740 CALL HCHAR(S,8,COMP)
750 CALL HCHAR(5,T,COMP)
760 NEXT COMP
770 PRINT TAB(17);"X=    Y=    "
780 REM *****************
790 REM ENTREZ L'ABSCISSE
800 REM *****************
810 A$="DEPART "
820 LIG=23
830 COL=3
840 GOSUB 1880
850 CALL HCHAR(23,22,30)
860 CALL HCHAR(23,27,30)
870 CALL SOUND(200,1600,0)
880 CALL KEY(0,K,S)
890 IF S<>1 THEN 880
900 IF K=1 THEN 2210
910 IF (K<49)+(K>55)THEN 880
920 CALL HCHAR(23,22,K)
930 CALL SOUND(200,1600,0)
940 CALL KEY(0,KE,ST)
950 IF ST<>1 THEN 940
960 IF KE=1 THEN 2210
970 IF (KE<49)+(KE>55)THEN 940
980 CALL HCHAR(23,27,KE)
990 A=VAL(CHR$(K))
1000 B=VAL(CHR$(KE))
1010 GOSUB 1480
1020 REM *****************
1030 REM ENTREZ L'ORDONNEE
1040 REM *****************
1050 A$="ARRIVEE"
1060 LIG=23
1070 COL=3
1080 GOSUB 1880
1090 CALL HCHAR(23,22,30)
1100 CALL HCHAR(23,27,30)
1110 CALL SOUND(200,1600,0)
1120 CALL KEY(0,K1,S1)
1130 IF S1<>1 THEN 1120
1140 IF K1=1 THEN 2210
1150 IF (K1<49)+(K1>55)THEN 1120
1160 CALL HCHAR(23,22,K1)
1170 CALL SOUND(200,1600,0)
1180 CALL KEY(0,KE1,ST1)
1190 IF ST1<>1 THEN 1180
1200 IF KE1=1 THEN 2210
1210 IF (KE1<49)+(KE1>55)THEN 1180
1220 CALL HCHAR(23,27,KE1)
1230 C=VAL(CHR$(K1))
1240 D=VAL(CHR$(KE1))
1250 GOSUB 1510
1260 GOSUB 1700
1270 REM *****************
1280 REM LE PION SE DEPLACE
1290 REM *****************
1300 CALL HCHAR(4+B*2,8+A*2,32)
1310 CALL HCHAR(4+D*2,8+C*2,122)
1320 IF A<>C THEN 1380
1330 IF KE1>KE THEN 1340 ELSE 1360
1340 CALL HCHAR(4+D*2-2,8+A*2,32)
1350 GOTO 2110
1360 CALL HCHAR(4+D*2+2,8+A*2,32)
1370 GOTO 2110
1380 IF K1>K THEN 1390 ELSE 1410
1390 CALL HCHAR(4+B*2,8+C*2-2,32)
1400 GOTO 2110
1410 CALL HCHAR(4+B*2,8+C*2+2,32)
1420 GOTO 2110
1430 REM *****************
1440 REM LE CARACTERE QUI
1450 REM SAUTE EST IL UN
1460 REM       PION ?
1470 REM *****************
1480 CALL GCHAR(4+B*2,8+A*2,CODE)
1490 IF CODE<>122 THEN 1630
1500 RETURN
1510 REM ******************
1520 REM    LE CARACTERE
1530 REM   D'ARRIVEE EST
1540 REM    IL UN VIDE??
1550 REM ******************
1560 CALL GCHAR(4+D*2,8+C*2,CODE1)
1570 IF CODE1<>32 THEN 1630
1580 IF (A<>C)*(B<>D)THEN 1630 ELSE 1590
1590 IF (A=C)*(B=D+2)THEN 1650
1600 IF (A=C)*(B=D-2)THEN 1650
1610 IF (A=C+2)*(B=D)THEN 1650
1620 IF (A=C-2)*(B=D)THEN 1650
1630 CALL SOUND(500,200,0)
1640 GOTO 810
1650 RETURN
1660 REM *****************
1670 REM LE CARACTERE SAUTE
1680 REM EST IL UN PION???
1690 REM *****************
1700 IF A<>C THEN 1780
1710 IF KE1>KE THEN 1720 ELSE 1750
```

```
1720 CALL GCHAR(4+D*2-2,8+C*2,COD)
1730 IF COD<>122 THEN 1630
1740 GOTO 1840
1750 CALL GCHAR(4+D*2+2,8+C*2,COD)
1760 IF COD<>122 THEN 1630
1770 GOTO 1840
1780 IF K1>K THEN 1790 ELSE 1820
1790 CALL GCHAR(4+D*2,8+C*2-2,COD)
1800 IF COD<>122 THEN 1630
1810 GOTO 1840
1820 CALL GCHAR(4+D*2,8+C*2+2,COD)
1830 IF COD<>122 THEN 1630
1840 RETURN
1850 REM ********
1860 REM PRINT AT
1870 REM ********
1880 REM
1890 FOR CARAC=1 TO LEN(A$)
1900 B$=SEG$(A$,CARAC,1)
1910 CALL HCHAR(LIG,COL+CARAC,ASC(B$))
1920 NEXT CARAC
1930 RETURN
1940 REM ************
1950 REM TRANSPARENCE
1960 REM ************
1970 FOR I=1 TO 16
1980 CALL COLOR(I,1,1)
1990 NEXT I
2000 RETURN
2010 REM ***************
2020 REM ROUTINE COULEUR
2030 REM ***************
2040 FOR I=1 TO 16
2050 CALL COLOR(I,2,1)
2060 NEXT I
2070 RETURN
2080 REM *****
2090 REM SCORE
2100 REM *****
2110 SCORE=SCORE+10
2120 A$=STR$(SCORE)
2130 LIG=1
2140 COL=28
2150 GOSUB 1880
2160 IF SCORE=310 THEN 2210
2170 GOTO 810
2180 REM ****************
2190 REM FIN DE LA PARTIE
2200 REM ****************
2210 CALL CLEAR
2220 PRINT "SCORE=";SCORE
2230 PRINT : : : : : : : : :
2240 PRINT "IL VOUS RESTE:";32-SCORE/10;"
 PIONS"
2250 IF SCORE<280 THEN 2280
2260 PRINT "VOUS AVEZ TRES BIEN JOUE!!!!"
2270 GOTO 2290
2280 PRINT "CE N'EST PAS UNE TRES BONNE P
ERFORMANCE!!!!": : :"MAIS VOUS FEREZ MI
EUX LA     PROCHAINE  FOIS!!!!!"
2290 PRINT "VOULEZ-VOUS REJOUER?(O/N)"
2300 CALL KEY(0,TOUCHE,ETAT)
2310 IF ETAT=0 THEN 2300
2320 IF (TOUCHE<>79)*(TOUCHE<>78)THEN 230
0
2330 IF TOUCHE=79 THEN 400
2340 CALL CLEAR
2350 END
2360 GOSUB 1970
2370 A$="JEU  DU"
2380 LIG=6
2390 COL=12
2400 GOSUB 1880
2410 FOR LIRE=1 TO 58
2420 READ A1,A2,A3,A4
2430 CALL HCHAR(A1,A2,A3,A4)
2440 NEXT LIRE
2450 A$="UNE TOUCHE POUR CONTINUER"
2460 LIG=24
2470 COL=3
2480 GOSUB 1880
2490 GOSUB 2040
2500 CALL KEY(0,KEY,STATUS)
2510 IF STATUS=0 THEN 2500
2520 CALL CLEAR
2530 GOSUB 1970
2540 PRINT "LE JEU CONSISTE A FAIRE       D
ISPARAITRE TOUTES LES       FICHES.":"LE
S FICHES SE DEPLACENT  "
2550 PRINT "HORIZONTALEMENT OU VERTI-   C
ALEMENT ET PRENNENT LE      PION A CONDI
TION QU'IL Y AIT"
2560 PRINT "UN TROU POUR SE DEPLACER,   C
OMME LE JEU DE DAMES.": :"CELLE QUI VIE
NT D'ETRE      "
2570 PRINT "SAUTEE EST ENLEVEE. LE JEU  E
ST TERMINE LOSQU'IL NE      RESTE QU'UNE
 FICHE": :
2580 PRINT "NEANMOINS, SI VOUS N'AVEZ   P
LUS DE POSSIBILITE DE       SAUTER, ARRE
TEZ LE JEU EN   "
2590 PRINT "FAISANT FCTN 7(AID)."
2600 PRINT : :"UNE TOUCHE POUR CONTINUER"
2610 GOSUB 2040
2620 CALL KEY(0,TOU,ETA)
2630 IF ETA=0 THEN 2620
2640 RETURN
2650 END
2660 DATA 1,1,130,3,2,1,130,1,3,1,130,3,4
,3,130,1,5,1,130,3,5,5,130,3,6,5,130,1,
7,5,130,1
2670 DATA 8,5,130,1,9,5,130,3,6,7,130,1,7
,7,130,1,8,7,130,1,9,9,130,1,10,9,130,1
,11,9,130,1
2680 DATA 12,9,130,1,13,9,130,3,13,13,130
,1,14,13,130,1,15,13,130,1,16,13,130,1,
17,13,130,1,11,13,130,1
2690 DATA 17,15,130,3,18,16,130,1,19,16,1
30,1,19,16,130,1,20,16,130,1,21,16,130,
1,13,19,130,3
2700 DATA 14,19,130,1,14,21,130,1,15,19,1
30,3,16,19,130,1,16,21,130,1,17,19,130,
1,17,21,130,1
2710 DATA 9,23,130,1,10,23,130,1,11,23,13
0,1,12,23,130,1,13,23,130,1,7,23,130,1
2720 DATA 5,25,130,3,6,25,130,1,6,27,130,
1,7,25,130,2,7,27,131,1,8,25,130,1,8,27
,130,1,9,25,130,1
2730 DATA 9,27,130,1,1,29,130,3,2,29,130,
1,3,29,130,2,4,29,130,1,5,29,130,3
```

Claustrophobia

Julien Thomas

Nos amis belges du club TISOFT-HOME nous envoient encore un très bon programme, écrit en **Basic TI**.

Le but de ce jeu est d'arriver à encercler un horrible monstre, à l'aide de briques disposées au hasard sur l'écran. Vous êtes représenté par une petite "bestiole" ni plus ni moins horrible que la chose à encercler. Les déplacements se font avec les touches fléchées, ce qui permet de déplacer les briques. Quand le monstre ne pourra plus bouger, vous devrez appuyer rapidement sur la touche "C". Le jeu semble simple mais l'auteur a ajouté quelques astuces qui compliquent les choses :

- les briques vertes sont fixes;
- de temps en temps, le monstre disparaît et réapparaît à un tout autre endroit de façon imprévisible. Le choix du niveau de jeu conditionne la fréquence des disparitions;
- si vous vous en approchez trop, le monstre peut vous dévorer.

Signalons enfin que le programme permet de fixer le nombre de briques à disposer sur l'écran. En choisissant un niveau de jeu élevé, il devient très difficile de "coincer la petite bête". Amusez-vous bien...

■

```
100 REM ******************
110 REM * CLAUSTROPHOBIA  *
120 REM ******************
130 REM *    BASIC TI     *
140 REM ******************
150 REM *    COPYRIGHT    *
160 REM *   TISOFT-HOME   *
170 REM * ET  99 Magazine *
180 REM ******************
190 CALL CLEAR
200 CALL SCREEN(5)
210 CALL CHAR(128,"007E7E7E7E7E7E00")
220 CALL COLOR(13,10,5)
230 CALL SOUND(500,262,0,523,0)
240 CALL HCHAR(3,3,128,3)
250 CALL VCHAR(4,3,128,4)
260 CALL HCHAR(7,4,128,2)
270 CALL SOUND(500,294,0,587,0)
280 CALL VCHAR(9,6,128,5)
290 CALL HCHAR(13,7,128,2)
300 CALL SOUND(500,330,0,659,0)
310 CALL HCHAR(1,9,128,3)
320 CALL VCHAR(2,9,128,4)
330 CALL VCHAR(2,11,128,4)
340 CALL HCHAR(3,10,128)
350 CALL SOUND(500,349,0,698,0)
360 CALL VCHAR(2,15,128,5)
370 CALL VCHAR(2,17,128,5)
380 CALL HCHAR(6,16,128)
390 CALL SOUND(500,392,0,784,0)
400 CALL HCHAR(8,13,128,3)
410 CALL VCHAR(9,13,128,2)
420 CALL HCHAR(10,14,128,2)
430 CALL VCHAR(11,15,128,2)
440 CALL HCHAR(12,13,128,2)
450 CALL SOUND(500,440,0,880,0)
460 CALL HCHAR(6,21,128,3)
470 CALL VCHAR(7,22,128,4)
480 CALL SOUND(500,494,0,988,0)
490 CALL HCHAR(2,26,128,3)
500 CALL VCHAR(3,26,128,4)
510 CALL VCHAR(3,28,128,2)
520 CALL VCHAR(4,27,128,2)
530 CALL HCHAR(6,28,128)
540 CALL SOUND(500,523,0,131,0)
550 CALL HCHAR(10,27,128,3)
560 CALL VCHAR(11,27,128,4)
570 CALL VCHAR(11,29,128,4)
580 CALL HCHAR(14,28,128)
590 CALL SOUND(500,587,0,147,0)
600 CALL HCHAR(17,3,128,3)
610 CALL VCHAR(18,3,128,4)
620 CALL HCHAR(19,4,128)
630 CALL SOUND(500,659,0,165,0)
640 CALL HCHAR(15,9,128,3)
650 CALL VCHAR(16,9,128,4)
660 CALL VCHAR(16,11,128,4)
```

```
670 CALL HCHAR(19,10,128)
680 CALL SOUND(500,698,0,175,0)
690 CALL HCHAR(16,16,128,3)
700 CALL VCHAR(17,16,128,4)
710 CALL HCHAR(17,18,128)
720 CALL HCHAR(18,17,128,3)
730 CALL HCHAR(19,19,128)
740 CALL HCHAR(20,17,128,3)
750 CALL SOUND(500,784,0,196,0)
760 CALL VCHAR(14,22,128,5)
770 CALL SOUND(500,880,0,220,0)
780 CALL HCHAR(17,26,128,3)
790 CALL VCHAR(18,26,128,4)
800 CALL VCHAR(18,28,128,4)
810 CALL HCHAR(19,27,128)
820 CALL SOUND(500,988,0,247,0)
830 CALL SOUND(500,1047,0,262,0)
840 FOR A=1 TO 1000
850 NEXT A
860 CALL CLEAR
870 PRINT TAB(8);"CLAUSTROPHOBIA"
880 PRINT TAB(7);"================"
890 PRINT :"ESSAYEZ DE CAPTURER L'ENNEMIE
N L'ENCERCLANT AVEC LES    BRIQUES."
900 PRINT "VOUS POUVEZ POUSSER HORIZON-TA
LEMENT OU VERTICALEMENT"
910 PRINT "UNE OU PLUSIEURS BRIQUES.   MA
IS PRENEZ GARDE !!!"
920 PRINT "LES BRIQUES VERTES NE       PE
UVENT PAS ETRE DEPLACEES"
930 PRINT "ET L'ENNEMI PEUT BRUSQUEMENTCH
ANGER DE POSITION."
940 PRINT "UTILISEZ LES FLECHES POUR   VO
US DEPLACER ET APPUYEZ SUR"
950 PRINT "'C' QUAND VOUS AUREZ        CA
PTURE L'ENNEMI. (FAITES LERAPIDEMENT)."
960 PRINT : :"POUR CONTINUER, APPUYEZ SUR
 N'IMPORTE QUELLE TOUCHE..."
970 CALL KEY(0,K,S)
980 IF S=0 THEN 970
990 CALL CLEAR
1000 PRINT "LE JEU OFFRE LA POSSIBILITE D
E CHOISIR ENTRE 10 NIVEAUX."
1010 PRINT :"DEUX POINTS VARIENT SELON LE
NIVEAU:":
1020 PRINT :"1) NOMBRE DE BRIQUES VERTES.
2) FREQUENCE DES CHANGEMENTS   DE POSIT
ION DE L'ENNEMI."
1030 PRINT :"IL EXISTE AUSSI UN NIVEAU
POUR LES ENFANTS (NIVEAU 0)."
1040 PRINT :"DANS CE CAS, LES POINTS 1 ET
2 NE S'APPLIQUENT PAS."
1050 PRINT :"BONNE CHANCE !!!"
1060 PRINT : : :"POUR COMMENCER, APPUYEZ
SUR N'IMPORTE QUELLE TOUCHE..."
1070 CALL KEY(0,K,S)
1080 IF S=0 THEN 1070
1090 CALL CLEAR
1100 PRINT "NIVEAU DE DIFFICULTE"
1110 PRINT
1120 INPUT "(0-10)               ":DIF
1130 PRINT
1140 PRINT
1150 IF (DIF>10)+(DIF<0)THEN 1100
1160 PRINT "NOMBRE DE BRIQUES:"
1170 PRINT
1180 INPUT "(MAX.400)            ":QUA
1190 IF (QUA>400)+(QUA<4)THEN 1130
1200 CALL CLEAR
1210 HYPER=0
1220 TIME=0
1230 IF DIF=0 THEN 1260
1240 TEST=(11-DIF)*10
1250 GOTO 1270
1260 TEST=1000
1270 CALL CHAR(136,"FFFFFFFFFFFFFFFF")
1280 CALL CHAR(137,"007E7E7E7E7E7E00")
1290 CALL CHAR(144,"99997E3C3C7E9981")
1300 CALL CHAR(145,"C3243C7F7F3C24C3")
1310 CALL CHAR(146,"81997E3C3C7E9999")
1320 CALL CHAR(147,"C3243CFEFE3C24C3")
1330 CALL CHAR(152,"3C7EFBFFF8F0793E")
1340 CALL CHAR(96,"18187E7E18181818")
1350 CALL COLOR(14,3,5)
1360 CALL COLOR(15,2,5)
1370 CALL COLOR(16,12,5)
1380 CALL CLEAR
1390 CALL HCHAR(1,2,136,30)
1400 CALL VCHAR(2,2,136,22)
1410 CALL VCHAR(2,31,136,22)
1420 CALL HCHAR(24,2,136,30)
1430 RANDOMIZE
1440 IF DIF=0 THEN 1520
1450 FOR A=1 TO 2*DIF
1460 X=INT(RND*22)+2
1470 Y=INT(RND*24)+5
1480 CALL GCHAR(X,Y,C)
1490 IF C=137 THEN 1460
1500 CALL HCHAR(X,Y,137)
1510 NEXT A
1520 FOR A=1 TO QUA
1530 X=INT(RND*22)+2
1540 Y=INT(RND*24)+5
1550 CALL GCHAR(X,Y,C)
1560 IF C<>32 THEN 1530
1570 CALL HCHAR(X,Y,128)
1580 NEXT A
1590 RICH=4
1600 MX=INT(RND*22)+2
1610 MY=30
1620 CALL HCHAR(MX,MY,152)
1630 X=INT(RND*22)+2
1640 Y=3
1650 CALL HCHAR(X,Y,145)
1660 FOR A=1 TO 3
1670 CALL SOUND(100,523,0)
1680 CALL SOUND(100,659,0)
1690 NEXT A
1700 CALL SOUND(400,523,0)
1710 CALL KEY(0,KEY,STATUS)
1720 TIME=TIME+1
1730 IF STATUS=0 THEN 2660
1740 IF KEY=69 THEN 1790
1750 IF KEY=68 THEN 2040
1760 IF KEY=88 THEN 1810
1770 IF KEY=83 THEN 2020
1780 GOTO 2660
1790 DIR=-1
1800 GOTO 1820
1810 DIR=1
1820 CALL SOUND(30,500,0)
1830 TEL=0
1840 CALL GCHAR(X+DIR,Y,A)
1850 IF A=32 THEN 1900
1860 IF (A=136)+(A=137)+(A=152)THEN 1980
1870 X=X+DIR
1880 TEL=TEL+DIR
1890 GOTO 1840
1900 IF TEL=0 THEN 1930
1910 IF A=136 THEN 1980
1920 CALL HCHAR(X+DIR,Y,128)
1930 X=X-TEL+DIR
1940 CALL SOUND(50,-1,0)
1950 CALL HCHAR(X-DIR,Y,32)
1960 CALL HCHAR(X,Y,145+DIR)
1970 GOTO 2660
1980 X=X-TEL
1990 IF A=152 THEN 2250
2000 CALL SOUND(100,110,0)
2010 GOTO 2660
2020 DIR=-1
2030 GOTO 2050
2040 DIR=1
2050 CALL SOUND(30,500,0)
2060 TEL=0
2070 CALL GCHAR(X,Y+DIR,A)
2080 IF A=32 THEN 2130
2090 IF (A=136)+(A=137)+(A=152)THEN 2210
2100 Y=Y+DIR
2110 TEL=TEL+DIR
2120 GOTO 2070
2130 IF TEL=0 THEN 2160
2140 IF A=136 THEN 1980
2150 CALL HCHAR(X,Y+DIR,128)
2160 Y=Y-TEL+DIR
2170 CALL SOUND(50,-1,0)
2180 CALL HCHAR(X,Y-DIR,32)
2190 CALL HCHAR(X,Y,146-DIR)
2200 GOTO 2660
2210 Y=Y-TEL
2220 IF A=152 THEN 2340
2230 CALL SOUND(100,110,0)
2240 GOTO 2660
2250 X=X+DIR
2260 IF TEL=0 THEN 2320
2270 CALL SOUND(10,500,0)
2280 CALL SOUND(10,500,0)
2290 CALL HCHAR(X-DIR,Y,32)
2300 CALL HCHAR(X,Y,145+DIR)
2310 GOTO 2660
2320 CALL HCHAR(X-DIR,Y,32)
2330 GOTO 2430
2340 Y=Y+DIR
2350 IF TEL=0 THEN 2410
2360 CALL SOUND(10,500,0)
2370 CALL SOUND(10,500,0)
2380 CALL HCHAR(X,Y-DIR,32)
2390 CALL HCHAR(X,Y,146-DIR)
2400 GOTO 2660
2410 CALL HCHAR(X,Y-DIR,32)
2420 GOTO 2430
2430 CALL HCHAR(X,Y,96)
2440 DATA 500,262,375,262,125,262,500,262
,375,311,125,294,250,294,250,262,250,26
2,250,233,500,262
2450 RESTORE 2440
2460 FOR A=1 TO 11
2470 READ T
2480 READ F
2490 CALL SOUND(T,F,0)
2500 NEXT A
2510 CALL CLEAR
2520 PRINT TAB(10);"VOUS PERDEZ"
2530 FOR A=1 TO 12
2540 PRINT
2550 NEXT A
2560 PRINT TAB(4);"VOULEZ VOUS REJOUER ?
(O/N)"
2570 CALL KEY(0,K,S)
2580 IF (K<>78)*(K<>79)THEN 2570
2590 IF K=78 THEN 3360
2600 GOTO 1090
2610 CALL KEY(0,K,S)
2620 IF K=67 THEN 2930
2630 HYPER=HYPER+1
2640 IF HYPER>TEST THEN 3210
2650 RICH=INT(RND*4)+1
2660 ON RICH GOTO 2670,2800,2670,2800
2670 ZIN=INT(RICH-1.5)
2680 CALL GCHAR(MX+ZIN,MY,CH)
2690 IF CH<>32 THEN 2750
2700 CALL HCHAR(MX,MY,32)
2710 CALL SOUND(30,-7,0)
2720 MX=MX+ZIN
2730 CALL HCHAR(MX,MY,152)
2740 GOTO 1710
2750 IF (CH<144)+(CH>147)THEN 2610
2760 CALL HCHAR(MX,MY,32)
2770 CALL HCHAR(MX+ZIN,MY,152)
2780 CALL SOUND(1000,-7,0)
2790 GOTO 2450
2800 ZIN=-(INT(RICH-2.5))
2810 CALL GCHAR(MX,MY+ZIN,CH)
2820 IF CH<>32 THEN 2880
2830 CALL HCHAR(MX,MY,32)
2840 CALL SOUND(30,-7,0)
2850 MY=MY+ZIN
2860 CALL HCHAR(MX,MY,152)
2870 GOTO 1710
2880 IF (CH<144)+(CH>147)THEN 2610
2890 CALL HCHAR(MX,MY,32)
2900 CALL HCHAR(MX,MY+ZIN,152)
2910 CALL SOUND(1000,-7,0)
2920 GOTO 2450
2930 CALL GCHAR(MX-1,MY,N)
2940 CALL GCHAR(MX,MY+1,E)
2950 CALL GCHAR(MX+1,MY,S)
2960 CALL GCHAR(MX,MY-1,W)
2970 IF (N<>136)*(N<>128)THEN 3020
2980 IF (E<>136)*(E<>128)THEN 3020
2990 IF (S<>136)*(S<>128)THEN 3020
3000 IF (W<>136)*(W<>128)THEN 3020
3010 GOTO 3070
3020 N=0
3030 E=0
3040 S=0
3050 W=0
3060 GOTO 1710
3070 CALL HCHAR(MX,MY,96)
3080 FOR F=1000 TO 110 STEP -40
3090 CALL SOUND(-200,F,0)
3100 NEXT F
3110 FOR A=1 TO 1000
3120 NEXT A
3130 CALL CLEAR
3140 PRINT TAB(10);"VOUS GAGNEZ"
3150 PRINT
3160 PRINT TAB(10);"TEMPS =";TIME
3170 FOR A=1 TO 10
3180 PRINT
3190 NEXT A
3200 GOTO 2560
3210 CALL HCHAR(MX,MY,32)
3220 MX=INT(RND*22)+2
3230 MY=INT(RND*30)+2
3240 CALL GCHAR(MX-1,MY,N)
3250 CALL GCHAR(MX,MY-1,W)
3260 CALL GCHAR(MX,MY,CH)
3270 CALL GCHAR(MX,MY+1,E)
3280 CALL GCHAR(MX+1,MY,S)
3290 IF (CH<>32)+((N<>32)*(E<>32)*(S<>32)
*(W<>32))THEN 3220
3300 CALL HCHAR(MX,MY,152)
3310 FOR F=110 TO 1000 STEP 70
3320 CALL SOUND(-100,F,0)
3330 NEXT F
3340 HYPER=0
3350 GOTO 1710
3360 CALL CLEAR
3370 END
```

"DISPLAY" et "ACCEPT" avec le module "PRK"

Sophie Ehster

Une importante découverte sur le module "Personal record keeping" (PRK), que l'on trouve en version française sous le nom de "gestion de fichiers".
Heureux les possesseurs de "PRK", puisqu'ils disposent de deux nouvelles instructions accessibles à partir du TI Basic. En effet, lorsque le module est connecté à la console, on peut se servir de deux nouveaux sous-programmes dont les rôles sont sensiblement les mêmes que ceux des instructions "ACCEPT AT" et "DISPLAY AT" du Basic Etendu.

CALL A

Le premier s'appelle tout simplement A. Il permet de saisir une donnée (comme le fait INPUT), mais en un point quelconque de l'écran et avec un contrôle optionnel des bornes de la variable. La syntaxe de cette instruction est la suivante :

CALL
A(Nligne,Ncolonne,Longmax,FCTNval,Variable,Min,Max)

chaque argument ayant la signification suivante :

- Nligne : numéro de la ligne (1 à 24)
- Ncolonne : numéro de la colonne (1 à 32)
- Longmax : longueur maximale de la variable saisie
- FCTNval : code de retour si la touche FCTN associée à certaines touches est pressée
- Variable : variable saisie
- Min : borne inférieure
- Max : borne supérieure

Codes de fonction retournés par A

- CLEAR (FCTN 4) : 2
- BEGIN (FCTN 5) : 6
- PROC'D (FCTN 6) : 5
- AID (FCTN 7) : 3
- REDO (FCTN 8) : 4
- BACK (FCTN 9) : 7
- ENTER : 1

Remarques :

1) Ces codes n'ont rien à voir avec ceux retournés par KEY qui correspondent, eux, aux valeurs ASCII des fonctions;

2) "CLEAR" sert à "nettoyer" la saisie en cours, le champ (dont la longueur est fixée par Longmax) est effacé, et le curseur est repositionné au début de celui-ci. Lorsque le champ est vide, et qu'on le valide par "ENTER", FCTNval vaut 2 (cf tableau). "CLEAR" ne permet pas, comme les autres fonctions, de sortir directement de la saisie. Il doit être validé par "ENTER";
Attention : "CLEAR" ne peut donc plus interrompre le programme lors de l'attente d'une saisie;

3) Une pression sur "ENTER" retourne (dans FCTNval) une valeur de 1 (à moins que le champ ne soit vide, comme on vient de le voir). Ainsi, lorsqu'on rentre une variable (numérique ou chaîne) que l'on valide par "ENTER", FCTNval vaut également 1;

4) Une variable numérique en cours de saisie n'est pas modifiée si l'on presse "ENTER" lorsque le champ est vide (action rejetée par "INPUT", qui écrit "*WARNING INPUT ERROR IN XXX").
Ainsi :

```
10 CALL CLEAR
20 A=66
30 CALL A(10,10,3,F,A)
40 PRINT A
```

écrira 66 (valeur inchangée), si on presse simplement ENTER;

5) Le contrôle du domaine de variation n'étant pas obligatoire, on peut omettre "Min" et "Max". S'ils sont présents, et que l'on tente d'entrer une variable hors de ces limites, un son grave est émis lorsque l'on presse "ENTER". La valeur n'est pas prise en compte, et le curseur se repositionne à son emplacement initial. Il est bien entendu impossible de spécifier des bornes lorsque A est utilisée pour la saisie de chaînes;

6) Lorsque la variable saisie est numérique, un son grave est émis si l'on presse une touche autre que 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, –, +, . ou E.

Exemple d'utilisation

CALL A(16,10,2,F,VAR,17,34)
permet de saisir la variable VAR sur 2 chiffres, à la seizième ligne et à la dixième colonne, avec un contrôle pour vérifier qu'elle est bien comprise entre 17 et 34. Si on presse "FCTN" (associée à une autre touche) au lieu de rentrer VAR, le code de la fonction sera stocké dans F.

CALL D

Le deuxième sous-programme se nomme D, et permet d'afficher une chaîne ou une variable, en un point quelconque de l'écran. Cette nouvelle instruction, tout comme la précédente, est très intéressante puisque le Basic TI ne possède pas le classique "PRINT AT", pourtant assez répandu par ailleurs.
La syntaxe est la suivante :

CALL
D(Nligne,Ncolonne,Longmax,Variable)

chacun de ces arguments ayant la même signification que pour A.

L'utilisation de ces deux nouveaux sous-programmes permet donc d'entrer une variable (numérique ou chaîne) en une position quelconque de l'écran, avec un message d'invitation, un contrôle de la saisie et enfin la possibilité de prendre une valeur par défaut.

```
10 CALL D(1,5,15,"Valeur de X :24")
20 CALL A(1,18,2,F,X,10,30)
```

positionne le curseur sur le 2 de 24, et attend l'introduction de X sur deux chiffres. Si l'on appuie sur "ENTER" sans rien écrire, X vaudra 24. Si l'on appuie sur "FCTN" et 7, alors F vaudra 3.

Ces deux instructions additionnelles permettent d'augmenter notablement l'ergonomie des programmes. Pour une exploitation de tableaux de données du type de celle de "PRK", ils sont indispensables.
De nombreuses ressources se cachent vraisemblablement au fond de plusieurs autres modules. A vous de les traquer, et n'oubliez pas de nous faire part de vos découvertes : elles intéressent tous les utilisateurs de TI-99.

■

Une erreur dans le manuel de la Mini-mémoire

Jean-Luc Bazanegue

A la page 72, consacrée à l'organisation de la mémoire morte de la Mini-mémoire, il est dit que la table des références prédéfinies est située à l'adresse >6F38. En fait, elle débute un peu plus bas, à l'adresse >6F0E. Rappelons au passage que chaque référence occupe huit octets, six pour les codes ASCII constituant le symbole (si le symbole utilise moins de six caractères, comme "ERR", les octets restant sont remplis par le caractère "espace") et deux octets pour l'adresse réelle. Vous trouverez ci-dessous le contenu de la table.

```
28430 >6F0E >55 >54 UTLTAB
28432 >6F10 >4C >54
28434 >6F12 >41 >42
28436 >6F14 >7020
28438 >6F16 >50 >41 PAD
28440 >6F18 >44 >20
28442 >6F1A >20 >20
28444 >6F1C >8300
28446 >6F1E >47 >50 GPLWS
28448 >6F20 >4C >57
28450 >6F22 >53 >20
28452 >6F24 >83E0
28454 >6F26 >53 >4F SOUND
28456 >6F28 >55 >4E
28458 >6F2A >44 >20
28460 >6F2C >8400
28462 >6F2E >56 >44 VDPRD
28464 >6F30 >50 >52
28466 >6F32 >44 >20
28468 >6F34 >8800
28470 >6F36 >56 >44 VDPSTA
28472 >6F38 >50 >53
28474 >6F3A >54 >41
28476 >6F3C >8802
28478 >6F3E >56 >44 VDPWD
28480 >6F40 >50 >57
```

```
28482 >6F42 >44 >20
28484 >6F44 >8C00
28486 >6F46 >56 >44 VDPWA
28488 >6F48 >50 >57
28490 >6F4A >41 >20
28492 >6F4C >8C02
28494 >6F4E >53 >50 SPCHRD
28496 >6F50 >43 >48
28498 >6F52 >52 >54
28500 >6F54 >9000
28502 >6F56 >53 >50 SPCHWT
28504 >6F58 >43 >48
28506 >6F5A >57 >54
28508 >6F5C >9400
28510 >6F5E >47 >52 GRMRD
28512 >6F60 >4D >52
28514 >6F62 >44 >20
28516 >6F64 >9800
28518 >6F66 >47 >52 GRMRA
28520 >6F68 >4D >52
28522 >6F6A >41 >20
28524 >6F6C >9B02
28526 >6F6E >47 >52 GRMWD
28528 >6F70 >4D >57
28530 >6F72 >44 >20
28532 >6F74 >9C00
28534 >6F76 >47 >52 GRMWA
28536 >6F78 >4D >57
28538 >6F7A >41 >20
28540 >6F7C >9C02
28542 >6F7E >53 >43 SCAN
28544 >6F80 >41 >4E
28546 >6F82 >20 >20
28548 >6F84 >000E
28550 >6F86 >58 >4D XMLLNK
28552 >6F88 >4C >4C
28554 >6F8A >4E >4B
28556 >6F8C >601C
28558 >6F8E >4B >53 KSCAN
28560 >6F90 >43 >41
28562 >6F92 >4E >20
28564 >6F94 >6020
28566 >6F96 >56 >53 VSBW
28568 >6F98 >42 >57
28570 >6F9A >20 >20
28572 >6F9C >6024
28574 >6F9E >56 >4D VMBW
```

```
28576 >6FA0 >42 >57
28578 >6FA2 >20 >20
28580 >6FA4 >6028
28582 >6FA6 >56 >53 VSBR
28584 >6FA8 >42 >52
28586 >6FAA >20 >20
28588 >6FAC >602C
28590 >6FAE >56 >4D VMBR
28592 >6FB0 >42 >52
28594 >6FB2 >20 >20
28596 >6FB4 >6030
28598 >6FB6 >56 >57 VWTR
28600 >6FB8 >54 >52
28602 >6FBA >20 >20
28604 >6FBC >6034
20606 >6FBE >44 >53 DSRLNK
28608 >6FC0 >52 >4C
28610 >6FC2 >4E >4B
28612 >6FC4 >6038
28614 >6FC6 >4C >4F LOADER
28616 >6FC8 >41 >44
28618 >6FCA >45 >52
28620 >6FCC >603C
28622 >6FCE >47 >50 GPLLNK
28624 >6FD0 >4C >4C
28626 >6FD2 >4E >4B
28628 >6FD4 >6018
28630 >6FD6 >4E >55 NUMASG
28532 >6FD8 >4D >41
28634 >6FDA >53 >47
28636 >6FDC >6040
28638 >6FDE >4E >55 NUMREF
28640 >6FE0 >4D >52
28642 >6FE2 >45 >46
28644 >6FE4 >6044
28646 >6FE6 >53 >54 STRASG
28648 >6FE8 >52 >41
28650 >6FEA >53 >47
28652 >6FEC >6048
28654 >6FEE >53 >54 STRREF
28656 >6FF0 >52 >52
28658 >6FF2 >45 >46
28660 >6FF4 >604C
28662 >6FF6 >45 >52 ERR
28664 >6FF8 >52 >20
28666 >6FFA >20 >20
28668 >6FFC >6050
```

Petites annonces (gratuites)

Vends (toutes propositions acceptées) ordinateur TI-99/4A, câble pour deux magnétophones, magnétocassette Thomson, manettes de jeu. Modules Munch Man, Parsec, The Attack, TI-Invaders, Gestion de fichier, Yahtzee, Teach yourself beginning Basic. Cassettes Oldies but Goldies et Teach yourself Extended Basic. Livres "à la découverte du TI-99", "Pratique du TI-99" et "Initiation au Basic". Docteur Blaisot-Le-Stanguennec - Tél : 473 00 51.

Vends Boîtier d'extension, carte 32K, carte série/par., contrôleur de disquettes, lecteur de disquettes, imprimante GP100 câble. Contacter le journal qui transmettra.

Cherche Modules Basic Etendu, TI-Invadeurs, Jeu rétro I et II. Gérard Lebon - Tél : (8) 322 84 23.

Achète module Basic Etendu. Monsieur Albert Saradin - 3, rue Michel Venini - Quincey - 70000 Vesoul.

Recherche lecteur de disquette pour TI-99. Possède contrôleur + boîtier d'extension + TI-Writer. Marcel Stoecklin - 2, Square des Acacias - 60400 Noyon - Tél : (4) 444 08 46.

Echangerais programmes et astuces. Cherche livre d'occasion (mais en bon état) sur le TI-99/4A. Cyril Lerat - 11, Rue de Lorraine - Vert Galand - 93410 Vaujours - Tél : 860 73 41.

Cherche module Mini-mémoire pour assembleur - Stanislas Blanchy -4, Rue Léo Delibes - 75116 Paris - Tél : (1) 704 43 17.

Achète d'urgence une carte contrôleur de disquette pour TI-99. Monsieur Marc Bayet - 1, Rue Neyret - 69283 Lyon Cedex 1.

Vends ou échange Editeur/Assembleur (jamais utilisé) et échange module Tombstone City.

Téléphoner à partir de 17 heures (sauf le lundi), Fernando Pico au 666 73 54.

Recherche des programmes de mathématiques, physique, chimie et gestion. Possibilité d'échange. Retour des documents si demandé. Je suis lycéen en 1ère S. Je cherche aussi des possesseurs de TI ou club informatique dans la région Charente Maritime. Philippe Boursier - 17, Rue de la Pierre Folle - 17270 Montguyon.

Recherche module Basic Etendu. David Cardon - Tél : (20) 29 06 21.

Echange ou vends des programmes et 3 modules de jeux (Hustle, Wumpus et Football) contre Basic Etendu, Mini-mémoire ou autres propositions. Rudy Hoevenaeghel - Rue Courte, 14 - 7780 Comines - Belgique.

Achète Basic Etendu, Blasto, Car Wars, Football, Scrabble. Alain - Tél : 525 34 18. ■

Réparez votre Assembleur ligne par ligne

Gilles Pavy

Si vous utilisez la Mini-mémoire, vous vous êtes sans doute rendus compte que les programmes NEW et OLD de l'assembleur ligne par ligne ne présentaient pas les différences annoncées dans le mode d'emploi. Afin d'y remédier, faites les modifications indiquées dans le listing. EASY BUG permet de réaliser ces modifications très simplement.

Sauvegardez votre nouvel assembleur sur cassette. Lorsque vous devrez interrompre la saisie d'un programme, vous pourrez couper l'alimentation du TI. Si, au moment où vous reprendrez votre travail, vous répondez OLD à la question PROGRAM NAME, vous retrouverez votre table des étiquettes ainsi que votre première adresse de travail. ■

```
Adresse  Code  Etiquette  Instruction  Operande  Commentaire
-------  ----  ---------  -----------  --------  -----------

>71A6    04E0  NEW        CLR          @>713E    a remplacer par CLR @>7CD6
>71A8    713E
>71AA    1002             JMP          >71B0
>71AC    0720  OLD        SETO         @>713E    a remplacer par SETO @>7CD6
>71AE    713E
>71B0    C80B             MOVE         R11,@>7182
>71B2    7182
   .
   .
   .
>7226    0560             INV          @>713E    a remplacer par INV @>7CD6
>7228    713E
>722A    1307             JEQ          >723A
>722C    0200             LI           R0,>7D00  a modifier si vous voulez chan-
>722E    7D00                                    ger l'adresse pointee par NEW
>7230    04E0             CLR          @>7190
>7232    7190
>7234    04E0             CLR          @>718E
>7236    718E
>7238    C240             MOV          R0,R9
>723A    0420             BLWP         @>79D2
>723C    79D2
>723E    7972             DATA         >7972
   .
   .
   .
```

Courrier des lecteurs

Alexandre Duback

Je vous écris afin que vous m'aidiez à résoudre le petit problème que voici : tous les programmes sont décalés vers la partie gauche du téléviseur, et une partie n'apparaît pas. De quoi cela provient-il, et que faire pour y remédier ?
A propos de "99 Magazine" : je le trouve très bon.

Eric Dauris - 13 Rue Hélène Boucher - 17300 Rochefort

Tous TI-99 provoquent ce décalage fort gênant, surtout si l'on programme en Assembleur ou en Pascal car, dans ce cas, la première colonne de caractères n'est pas visible. Il n'est pas possible (du moins pas simplement) d'agir sur l'ordinateur pour recadrer l'image. Par contre, les réglages du téléviseur peuvent être modifiés, afin d'obtenir un cadrage satisfaisant. N'importe quel technicien peut effectuer le réglage de votre récepteur en quelques minutes. Si vous faites réaliser ces réglages, les images télévisées seront décalées vers la droite, mais ceci n'est visible que dans des cas particuliers (mires, génériques ou images très géométriques).

Tout d'abord, je dois vous dire que je trouve "99 Magazine" très bien structuré et que les programmes y sont de très bonne qualité. Mais je vous écris surtout pour vous donner un autre tuyau au sujet des manettes de jeu (numéro 3 de "99").
J'ai eu moi aussi ce problème et je l'ai résolu de la manière suivante :

J'ai tout d'abord pensé à du papier de chocolat mais puisque ce "matériel" n'est pas auto-collant (seul le chocolat l'est...), j'ai eu l'idée de mettre des "gommettes" métallisées. C'est très économique, cela s'achète dans les librairies et c'est très facile à remplacer. J'espère que cela rendra service à tous les "Texans".

*Je voudrais aussi vous demander un tuyau. J'ai essayer de faire (en haute résolution) un graphique sur tout l'écran, mais mon dessin ne comportait que quelques "cases" identiques; tous les autres carrés avait un code ASCII différent. J'ai vite été à court de codes. Aussi, j'aimerais savoir s'il est possible de faire 768 (24*32) caractères différents et, si oui, comment.*

Jean-Pierre Diot - 47, Rue Jacquart - 51100 Reims

Nous vous remercions pour votre tuyau (et vos compliments) et nous espérons que l'article sur la haute résolution graphique, publié dans ce numéro, vous apportera quelques éclaircissements sur le sujet.

J'ai découvert récemment votre revue qui est une vraie mine d'or pour les utilisateurs du TI-99 puisqu'il est devenu extrêmement difficile de trouver, dans les magazines d'informatique individuelle, des programmes utilisables "tel quel", ou des idées intéressantes qui soient directement applicables au TI-99/4A. En effet, lorsqu'on débute dans cette discipline (dont on ignore tous les secrets) et que l'on essaie d'entrer un programmme non destiné au TI-99, on bute sur certaines instructions non prévues sur cet ordinateur, et le programme n'est pas exécutable, à cause de deux ou trois ligne mal adaptées.
J'aimerais savoir comment on peut remanier un programme quand on rencontre les instructions "POKE" ou "CLEAR" qui n'existent pas sur le TI, même en Basic Etendu.

Longue vie à votre revue !

François Palabaud - 5, Rue Grenette - 42190 Charlieu

Le Basic est un langage que l'on trouve sur tous les micro-ordinateurs, malheureusement, pratiquement chaque constructeur a son propre Basic, et l'adaptation des programmes n'est pas toujours chose aisée. L'instruction "POKE" est très courante et se trouve sur le TI sous le nom de "CALL LOAD", à condition de posséder le Basic Etendu, l'Editeur/Assembleur ou la Mini-mémoire. Cette instruction effectue le chargement d'une valeur directement dans une adresse de la mémoire vive. Donc, pour utiliser cette instuction dans une "traduction", il faut posséder une bonne connaissance de la structure des deux ordinateurs. En général, l'instruction "CLEAR" Provoque une réinitialisation (remise à zéro) de toutes les variables présentes dans le programme, et n'est pas disponible sur le TI-99. Par contre, il est parfaitement possible de la simuler :

- en Basic TI avec

```
150 A=0
160 B=0
170 C=0
etc...
```

- en Basic Etendu avec

```
150 A,B,C=0
etc...
```

Je tiens tout d'abord à vous féliciter et vous remercier pour la création d'un magazine destiné uniquement aux utilisateurs du TI-99 et la continuation de son édition malgré l'arrêt de fabrication de notre micro.
Il apparaît en effet qu'un magazine aussi spécialisé soit le seul investissement rentable pour l'utilisateur d'un micro, une fois son choix arrêté sur tel ou tel appareil. Je me réjouis donc que le TI-99 possède le sien.

J'en viens maintenant à mes questions :

- quels sont les éléments indispensables à l'accès au langage Assembleur ? D'après le manuel Basic Etendu, il semblerait que seule l'extension 32 Ko soit nécessaire. Est ce bien cela ?
- la brochure d'accompagnement du module Basic Etendu fait mention d'une option "Pre-scan" dont le but serait de supprimer l'analyse du programme après la commande "RUN", et d'éviter ainsi la perte de temps qui en résulte. Cependant, je ne retrouve pas cette option dans le manuel. Pouvez-vous m'indiquer le mode d'emploi de cette option ?

Pour terminer, je voudrais faire une ou deux remarques qui vous seront peut-être utiles.

Dans le numéro 2, Laurent Pecheux vous signalait une erreur dans le programme "Le dernier robot", en ligne 220. Il n'a en cela pas tout à fait tort. Il pourrait effectivement s'agir d'une erreur, mais d'une erreur par ommission. En effet, la ligne 220 a pour but de redéfinir les caractères après lecture en DATA du numéro et du descripteur du nouveau caractère. L'examen des lignes 5240 et 5250 fait ressortir que le programme redéfinit des caractères appartenant aux groupes 15 et 16 auxquels nous n'avons plus accès en Basic Etendu. Ainsi, si notre confrère a tenté de faire tourner le dernier robot en Basic Etendu, il est normal qu'il ait obtenu un message d'erreur et un

arrêt en ligne 220. Afin d'éviter ce genre de déboire, il serait utile que vous indiquiez, pour chaque programme en Basic TI, les incompatibilités avec le Basic Etendu, d'autant plus qu'elles sont relativement restreintes.

D'autre part, je pense qu'un article sur les erreurs, leurs messages et leur recherche pourrait être très intéressant pour les débutants. En effet, il n'y a rien de plus stressant qu'un programme qui bloque et de ne pouvoir y remédier. Il n'existe peut-être pas de recette pour rechercher une erreur, mais un petit truc comme l'utilisation de l'instruction "PRINT" permet très souvent de cerner la cause de l'erreur. Ainsi, Laurent Pecheux, une fois son programme bloqué en ligne 220, aurait pu tester la valeur des variables de cette ligne au moment du blocage par PRINT A et PRINT B$, et s'interroger sur l'exactitude et la possibilité des valeurs obtenues.

Christian Michel - 2, Rue F. Pelloutier - 38000 Grenoble

Pour accéder au langage Assembleur, il existe deux possibilités :

1) le module Editeur/Assembleur. Dans ce cas il faut disposer d'un lecteur de disquettes et de l'extension 32 Ko;

2) le module Mini-mémoire, avec lequel seule la console de base est nécessaire. L'assembleur ligne par ligne de la Mini-mémoire est beaucoup moins pratique que l'Editeur/Assembleur mais permet néanmoins de réaliser des petits programmes.

Il est en effet possible d'annuler partiellement l'action du "Pre-scan". La méthode à utiliser est indiquée dans le petit supplément fourni avec le manuel du Basic Etendu. Je ne dispose pas ici d'assez de place pour vous expliquer comment utiliser cette fonction. Nous pourrions développer cette particularité du Basic Etendu dans un prochain article mais, en attendant, vous pouvez consulter le programme de Gilbert Arribet baptisé "Sapin de Noël", publié dans le numéro 3, puisqu'il utilise cette fonction.

Nous avions mis en évidence, dans le courrier des lecteurs du numéro 3, les problèmes d'incompatibilité entre le Basic TI et le Basic Etendu, mais il n'était pas inutile d'y revenir. L'instruction "PRINT" est effectivement la plus efficace lorsqu'il s'agit de trouver la cause d'une erreur. Elle remplace avantageusement toutes les instructions de mise au point (TRACE, UNTRACE, etc...) dont l'utilisation est assez lourde.

Pouvez vous me dire si je peux me procurer votre magazine chez un marchand de journaux et, si oui, lequel ?

P. Denost - 35, rue Lamartine - 78000 Versailles

La revue "99 Magazine" est distribuée dans les boutiques informatiques et les librairies spécialisées par "PSI diffusion" mais n'est pas diffusée en kiosque.
D'autre part, vous êtes nombreux à vous être "cassé le nez" en essayant de venir nous voir, je profite donc de l'occasion pour signaler à nos lecteurs que le "49, rue Lamartine" est uniquement une adresse postale (en effet, les Editions MEV ne font, directement, que la vente par correspondance et le service des abonnements). ■

A vous de programmer

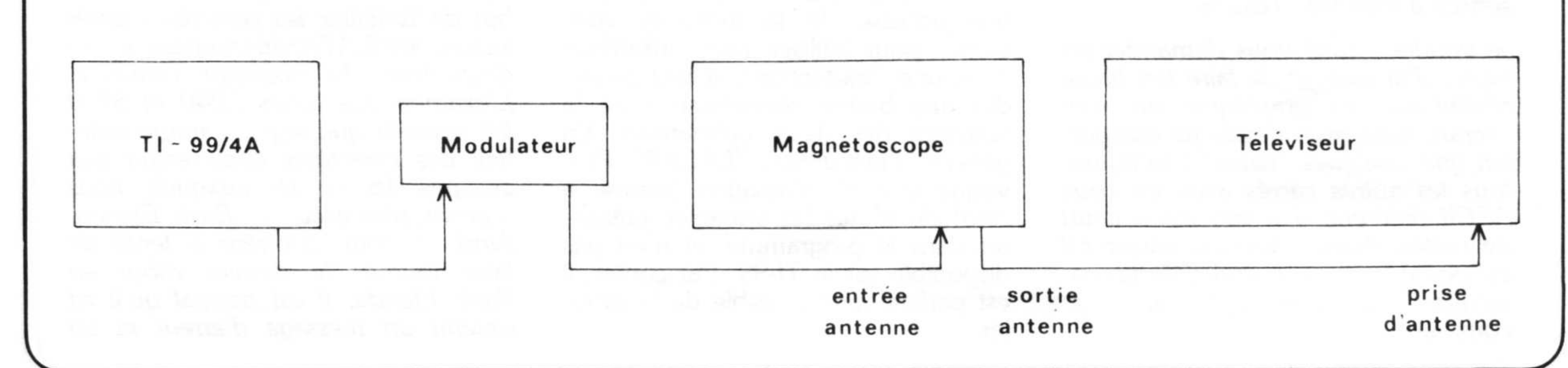
Si l'on en juge par la quantité de programmes simulant une horloge ou un chronomètre parvenus jusqu'à nos bureaux, il semble que cette rubrique soit appréciée.
Nous vous proposons donc à nouveau d'écrire un programme sur un thème donné. Comme par le passé, nous publierons le meilleur et son auteur recevra un abonnement pour quatre numéros avec cassettes.

Cette fois, il faudra écrire un programme ou un sous-programme capable de convertir des nombres entrés sous une forme numérique, en leurs équivalents "en toutes lettres". Par exemple :

1236861
deviendra :
UN MILLION DEUX CENTS TRENTE SIX MILLE HUIT CENTS SOIXANTE ET UN

Il est aussi envisageable de faire la conversion inverse, mais cela est un peu plus compliqué. Les lecteurs munis du Basic Etendu pourront créer un sous-programme à variables locales de manière à ce qu'il puisse être implanté dans n'importe quel autre logiciel.

Le fruit de votre travail devra nous parvenir avant le premier mai, sauf pour nos amis Canadiens auxquels nous accordons deux semaines supplémentaires. Le temps nécessaire à l'acheminement du courrier entre la France et le Canada fait que notre ancienne méthode était un peu injuste. ■



Des cassettes pour le TI-99/4A

Les cassettes A, B, C et D sont toujours disponibles et nous ajoutons à notre "catalogue" la cassette E. Voici une brève récapitulation du contenu des cassettes précédentes.

Cassette 99A

Basic TI :

Electronique
Mineur

Basic Etendu :

Prêt

Cassette 99B

Basic TI :

Yathzee
Division

Basic Etendu :

Chardef

Cassette 99C

Basic TI :

Division
Course

Basic Etendu :

Dames
Nim
Isola
Schmoo

Cassette 99D

Basic TI :

Caractérologie
Car driver
Cannibales

Basic Etendu :

Conjugaisons
Poker

Cassette 99E

Sur cette nouvelle cassette, vous trouverez six programmes, quatre en Basic TI et deux en Basic Etendu. Nous vous rappelons que les programmes écrits en Basic Etendu ne fonctionnent pas sans le module du même nom, et que les programmes écrits en Basic TI ne fonctionnent pas forcément en Basic Etendu.

Mic-Math

Il s'agit d'un jeu basé sur les mathématiques. Le programme, écrit en **Basic TI**, affiche un quadrillage dans lequel sont visualisées des opérations élémentaires. Il faut se déplacer à l'intérieur de ce quadrillage de façon à obtenir le résultat demandé par l'ordinateur. La solution doit être trouvée dans un temps limité et plusieurs niveaux de jeu sont disponibles.

Course de chevaux

Amis turfistes, bonjour... Ce programme en **Basic TI** accepte jusqu'à cinq joueurs. Chaque joueur mise sur un des cinq chevaux et, au terme de la course, celui qui a choisi la bonne bête empoche les gains.

Jeu de poursuite

Vous êtes un petit bonhomme et vous êtes poursuivi par un énorme globule affamé. Le but de ce jeu, écrit en **Basic TI**, est de manger le plus de carrés de couleurs avant que le temps accordé soit écoulé, à moins que vous soyez dévoré entre temps. Les points obtenus varient suivant la couleur des carrés. Les niveaux de jeu vont de très facile à impossible.

Jeu de dés

Ce jeu simule le jeu de dés tel qu'il se pratique sur les tables de jeu du Nevada (Craps). Ecrit en **Basic TI**.

Guerre atomique

Ce programme écrit en **Basic Etendu** est absolument inoffensif. Le but du jeu est de détruire six silos atomiques répartis sur le territoire ennemi. Ce territoire est une grille de 24 cases. Après chaque tir, vous serez renseigné sur le nombre de silos restants au Nord, au Sud, à l'Est et à l'Ouest de la case visée. Vous pourrez, si vous le désirez, effacer une rangée entière, si vous croyez qu'aucun silo ne s'y trouve.

Course en ligne droite

Il s'agit d'un jeu de stratégie écrit en **Basic Etendu**, dans lequel votre adversaire est le TI-99.

■

99 Magazine s'adresse...

... aux débutants

Vous venez d'acquérir un TI-99/4A. Tout est maintenant déballé et installé et vous êtes en train de découvrir qu'un ordinateur, ça ne marche pas tout seul... Il existe bien des brochures d'accompagnement, mais elles ne sont pas toujours rédigées de façon claire. En outre, le seul renseignement dont vous avez besoin en ce moment ne s'y trouve peut-être pas.

"99" vous apporte chaque trimestre de nombreux renseignements, des programmes complets, des exemples. Déjà, notre premier numéro comporte d'importants articles d'initiation au Basic ou de présentation de programmes.

De plus, "99" édite avec chaque numéro une cassette regroupant tous les programmes du numéro. Vous pouvez ainsi utiliser des programmes que vous y avez vus, même si vous n'avez pas complètement compris le listing. En effet, vous savez bien utiliser des jeux ou des programmes de gestion de fichiers sans pour autant être capables de comprendre intégralement le listing de ces programmes.

... aux habitués

Malgré l'apparition récente du TI-99 sur le marché européen, vous êtes déjà nombreux qui, habitués, maîtrisez bien l'un des Basics. Nous analysons pour vous et testons les nouveaux produits, logiciels et matériels. Nous vous donnons des idées pour résoudre des problèmes qui vous tracassaient depuis longtemps.

En étudiant les nombreux programmes que nous vous proposons, vous pourrez perfectionner votre connaissance du Basic simple et du Basic Etendu mais aussi, à partir des prochains numéros, vous initier au Pascal et à l'assembleur. Dans pratiquement chaque numéro de "99", vous trouverez des recettes ou des programmes pour mieux programmer et utiliser votre TI-99.

... aux experts

Vous êtes déjà quelques-uns à mériter le titre d'experts. Chacun d'entre vous connaît plus particulièrement un langage ou un domaine d'application. "99", en publiant les écrits d'autres experts, peut vous en apprendre plus, que ce soit dans votre domaine de prédilection ou dans d'autres domaines.

... à tous ses lecteurs

Quel que soit votre niveau, vous pouvez avoir des informations, des idées ou des programmes à transmettre aux autres. Faites nous les parvenir afin que nous puissions faire partager ce savoir que vous avez acquis. L'idéal est de nous envoyer une cassette avec vos programmes et un article dactylographié. N'oubliez pas d'indiquer votre adresse et votre numéro de téléphone !

SYBEX

Gagnez un Voyage à Silicon Valley

Oui, vous êtes invités gratuitement à gagner un voyage d'une semaine pour deux personnes au pays de la micro-informatique.

MICRO-EXPO, 9ᵉ congrès-exposition, carrefour international de la micro-informatique se tiendra à Paris, au Palais des Congrès du 22 au 26 mai 1984.

Visitez cette manifestation qui vous offrira la possibilité exceptionnelle de rencontrer et de dialoguer avec plus de 200 exposants français et étrangers, de suivre une trentaine de conférences professionnelles et grand public : comment choisir son tableur électronique, les systèmes intégrés : 1 - 2 - 3, Lisa, Visi/On, MS-WIN, choisir son micro, comptabilité et bases de données, Basic... Découvrez les dernières nouveautés dont certaines seront présentées en exclusivité.

La multiplicité et la diversité des produits et techniques présentés à ce grand rendez-vous annuel constitueront pour vous la garantie du bon investissement et de la bonne décision.



COUPON RÉPONSE A COMPLÉTER ET A RETOURNER A SYBEX - 6-8, impasse du Curé - 75018 PARIS

Celui-ci est votre titre de participation au tirage au sort qui aura lieu en présence de Maître P. Chale, huissier à Paris. Merci de m'adresser :

☐ une entrée gratuite et le programme détaillé des conférences.

☐ un passeport valeur 100 F T.T.C. me donnant droit à l'entrée permanente au salon, au guide de la micro 84 et de participer à toutes les conférences de mon choix (attention le nombre de places est limité !).

NOM ....

Prénom ....

Société ....

N° .... Rue ....

Code postal .... Ville ....

Activité de l'entreprise ....

Fonction ....

Ci-joint chèque de 100 F.



lisez

L'ORDINATEUR INDIVIDUEL

Vous y trouverez :

• l'actualité et les tendances de l'informatique individuelle • les bancs d'essais des principaux matériels • des panoramas et des tests comparatifs • le point des grandes manifestations internationales • des articles d'initiation • des synthèses • des programmes • des interviews "exemplaires" • des conseils • des idées • des astuces

L'ORDINATEUR INDIVIDUEL, chez votre marchand de journaux